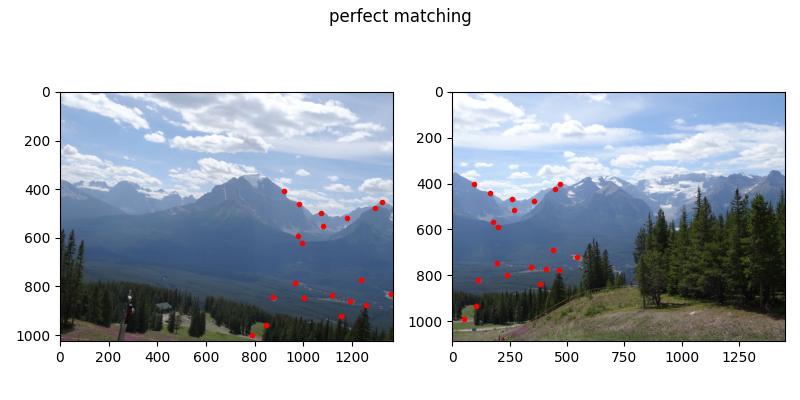
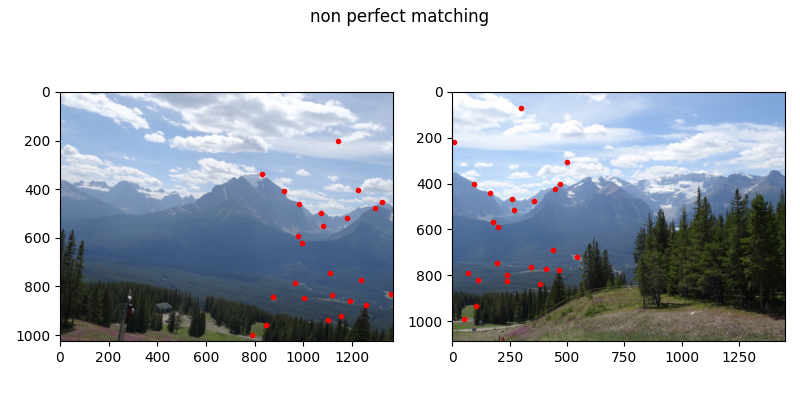
Loading the data:





Part A:

Homography is the projection operator between two sets of homogenous coordinates, of the form:

Where are a pair of matching points. Denote the rows of by for =1, 2, 3, then:

By rearranging equations (1) and (2) we get:

By noticing that we have 9 unknowns and each pair of matching points gives us 2 constrains, we need at least 4 matching points (not more because it is up to scale). In our case we have 20 pairs of matching/corresponding points (it may improve accuracy), so we have an overdetermined system of equations. From matches\_perfect.mat we take the matching points (source image) match\_p\_src, (target/destination image) match\_p\_dst.

To approximate the homography, we need to approximate the solution to .

Using SVD, decompose , and the last column of is an approximate solution to the least squares problem .

3.

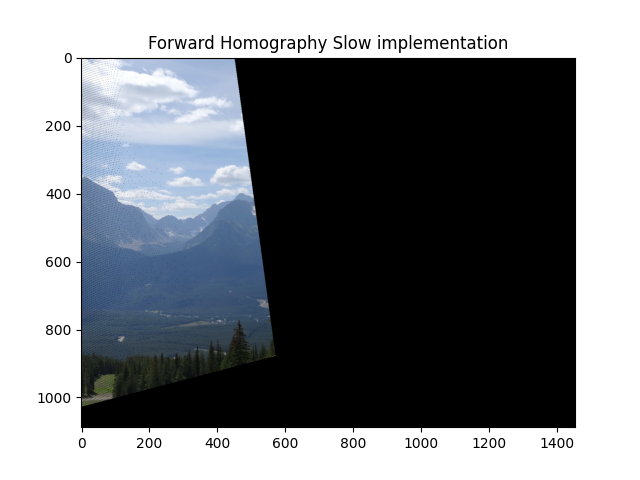
Naive Homography 0.0003 sec

[[ 1.43457211e+00 2.10443228e-01 -1.27718676e+03]

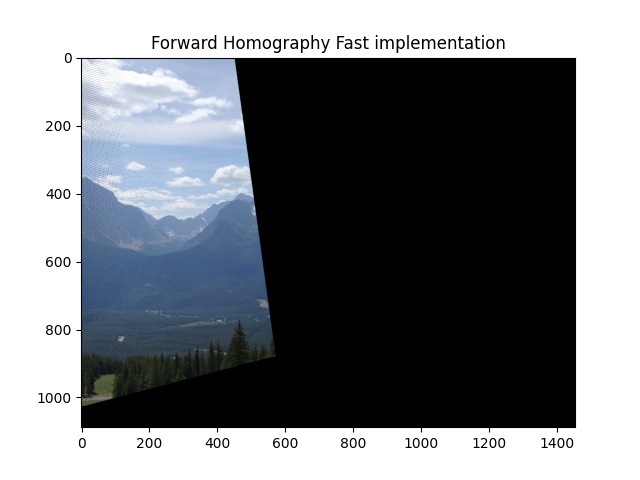
[ 1.34265015e-02 1.34706121e+00 -1.60455744e+01]

[ 3.79279274e-04 5.56523173e-05 1.00000000e+00]]

4.



5.

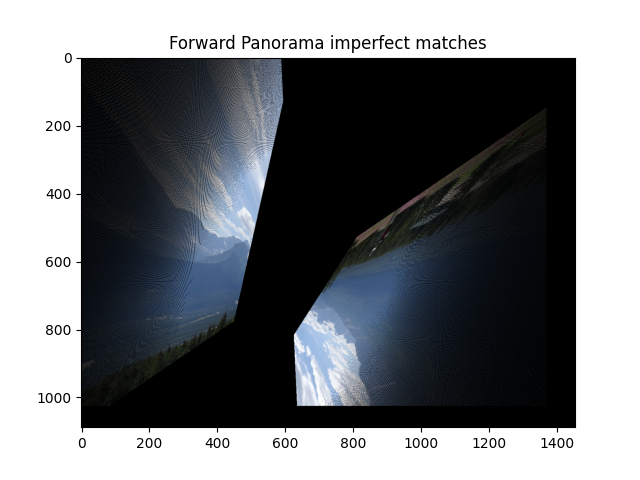


6.

The main problem is that after applying the transformation matrix on the source coordinates, most of the points are mapped to points exceeding the bounding dimensions of the destination image. Also, the resulting coordinates are not discrete, and after rounding it may be that different pixels in the source image are mapped to the same location in the destination, and some pixels in the destination image remain empty. This appears as some black lines and holes on the resulting image (can be seen when zoomed in).

7.

The Result is different and not the expected one: it is flipped and rotated. Thus wrong correspondences (outliers) provoke a bad result, that is because compute\_homographt\_naive (which computes transformation matrix H) cannot deal with the error provided from the given outliers and gives unreliable H.



Part B

10.

In class we derived:

In the Homography model . Given , for we get the smallest number of iterations is:

And for :

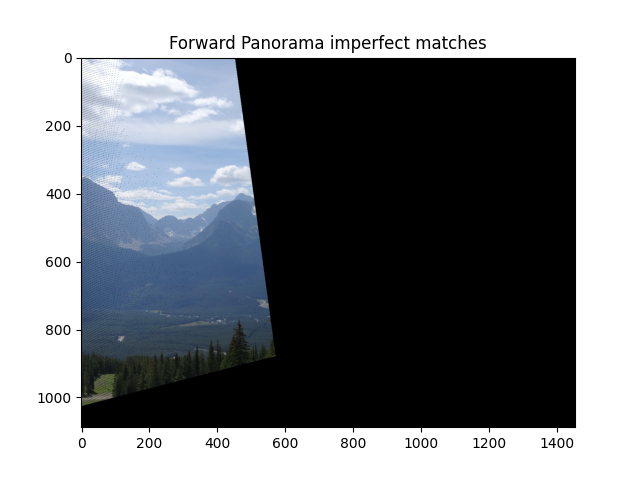
In each iteration we sample 4 points, so to make sure we cover all options we need to change the algorithm so that randomizations are not repeated. Then, total number of unique randomizations:

12.

|  |  |
| --- | --- |
| RANSAC Homography 0.0706 sec  [[ 1.43457211e+00 2.10443228e-01 -1.27718676e+03]  [ 1.34265015e-02 1.34706121e+00 -1.60455744e+01]  [ 3.79279274e-04 5.56523173e-05 1.00000000e+00]] | Naive Homography 0.0009 sec  [[ 1.43457211e+00 2.10443228e-01 -1.27718676e+03]  [ 1.34265015e-02 1.34706121e+00 -1.60455744e+01]  [ 3.79279274e-04 5.56523173e-05 1.00000000e+00]] |

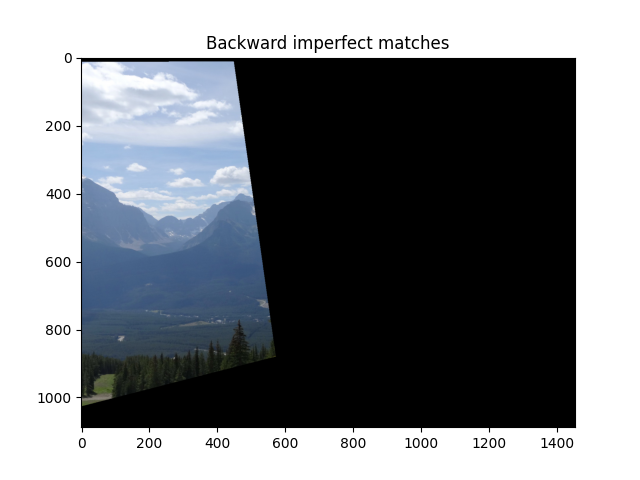
RANSAC Homography Test : [0.8, 1.8406513693149382]

The values of the RANSAC homography are very close to the naïve homography computed using perfect matches only. This means the RANSAC algorithm correctly found the mismatching points and ignored them in finding H. compared to section 7 we got much better the results, the image is indistinguishable from the result achieved using only perfect matches:



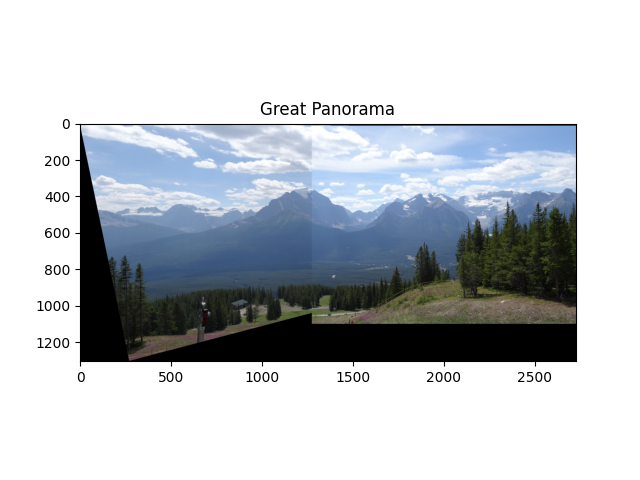
Part C:

13.



Here we can see there are no holes and no aliasing, as we saw in section 10 (due to rounding off in the forward transform).

16.



17.

