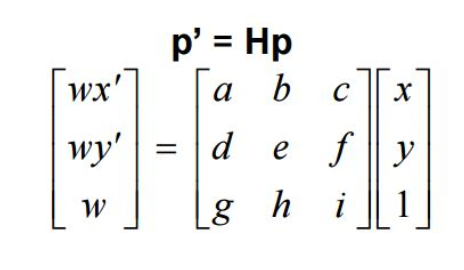
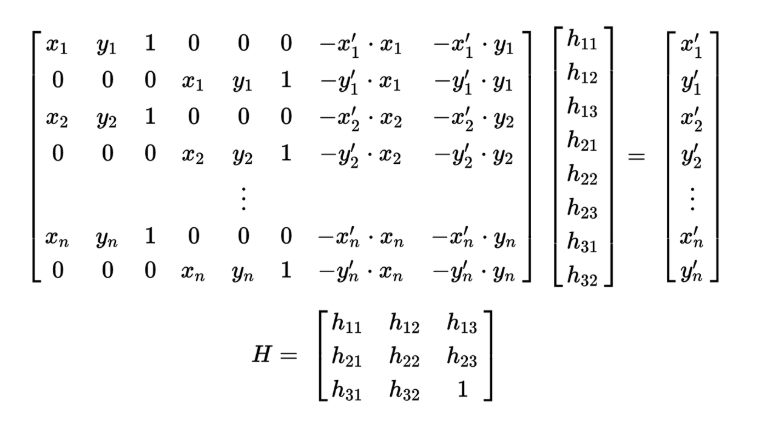
1. We would like to solve:



Where p are the source points and p’ are the destination points. We can solve for the homography matrix as follows:



We will call the vector on the right hand side M2 (containing the destination points) and the matrix on the left hand side M1 and use the least squares methodology to find H as follows:

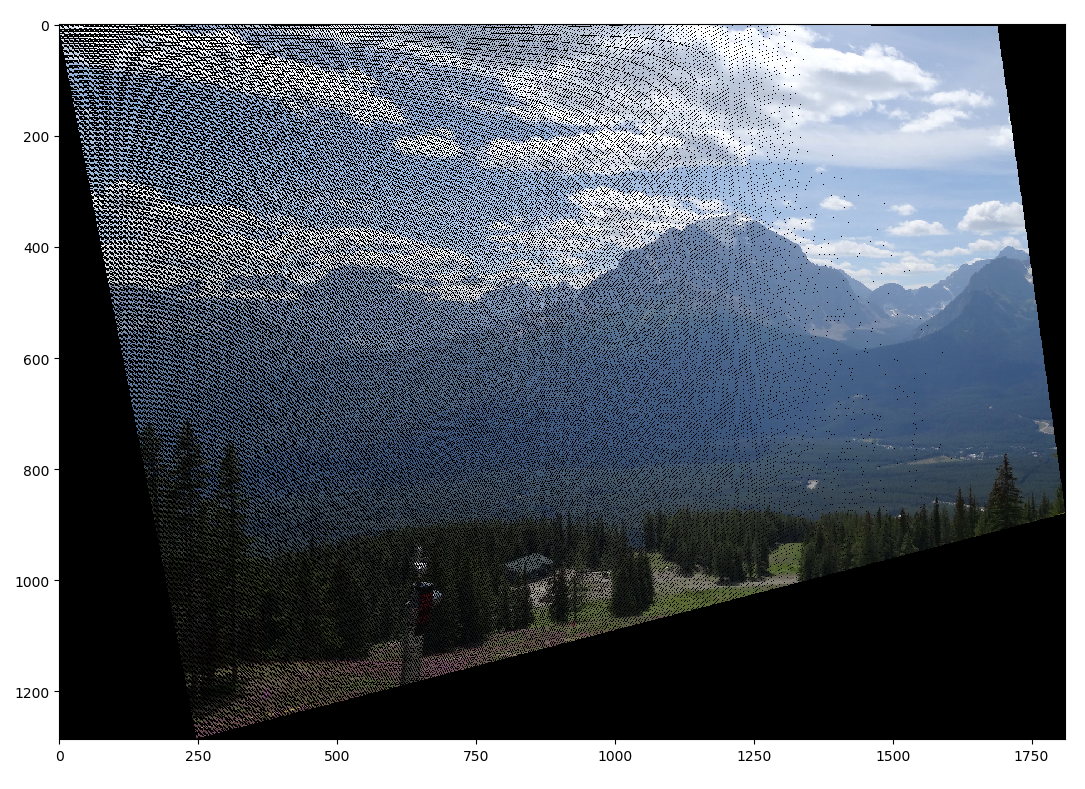
H = ( M1 transpose \* M1 )inv \* ( M1 transpose \* M2)

1. See code
2. H = [ 1.39472444e+00 2.01438863e-01 -1.23828556e+03]

[ 3.84188164e-03 1.30716833e+00 -7.80395273e-01]

[ 3.55783803e-04 4.19398052e-05 1.00000000e+00]

1. The image received is :



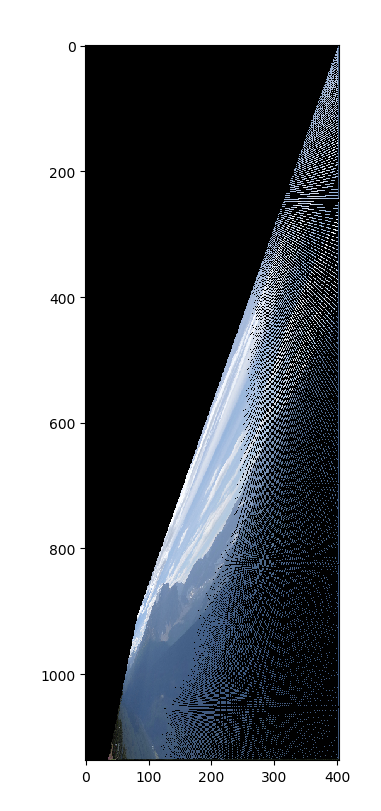
1. The transformation is not smooth meaning we have “holes” in the target image because the range of the transformation appears to be larger than the domain so that the number pixel transformed is not sufficient. The shape of the new image is not rectangular. Also the holes appears to be located mostly on the left side and are not evenly spread.

Our New H (calculated using “matches” containing outliers):

[-3.21938547e-02 -5.88287579e-02 1.02051638e+02]

[-3.95818251e-01 4.47497305e-02 4.82946288e+02]

[-6.30013633e-04 -2.49197318e-04 1.00000000e+00]

We got the following image mapping:

We can see that most of the pixels were not mapped to the valid pixel range and therefore the target image is significantly smaller the “holes” artifact is also apparent here and was even increased.

**PART B**

1. See code
2. See code

K – number of needed randomizations to guarantee the needed level of confidence.

P - success confidence guarantied

W – inliers percent

n – minimum number of points required to compute the homography matrix.

For n= 4 and w = 0.8 , p= 0.9 we get k ~ 5

For n= 4 and w = 0.8 , p= 0.99 we get k ~ 9

In order to cover all possibilities for say 30 data points while in each iteration we choose 4 points we use “m choose 30”:

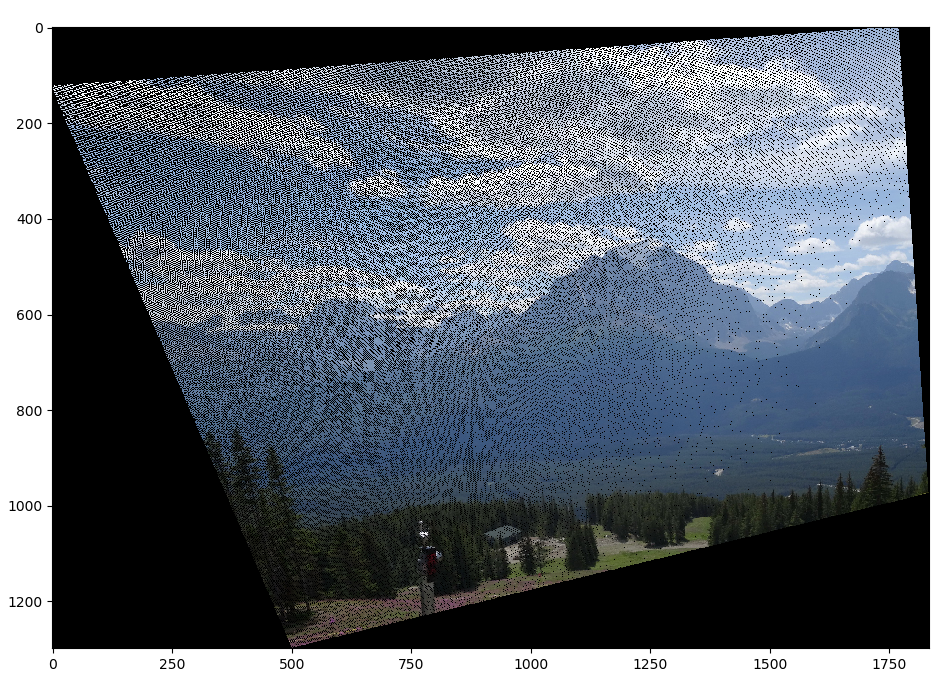
1. Received best H:

[ 1.40758885e+00 2.52105630e-01 -1.27402138e+03]

[-1.14219346e-01 1.51415480e+00 1.98605308e+01]

[ 2.26128168e-04 3.09585640e-04 1.00000000e+00]

Homography image obtained with RANSAC for n = 10:



We can notice that in comparison to the computation with outliers without implementing RANSAC (6) we got a significantly improved result seeing as the outliers we mostly neglected while calculating H\_RANSAC and therefore our result is much similar to the naïve forward mapping seen in 4.