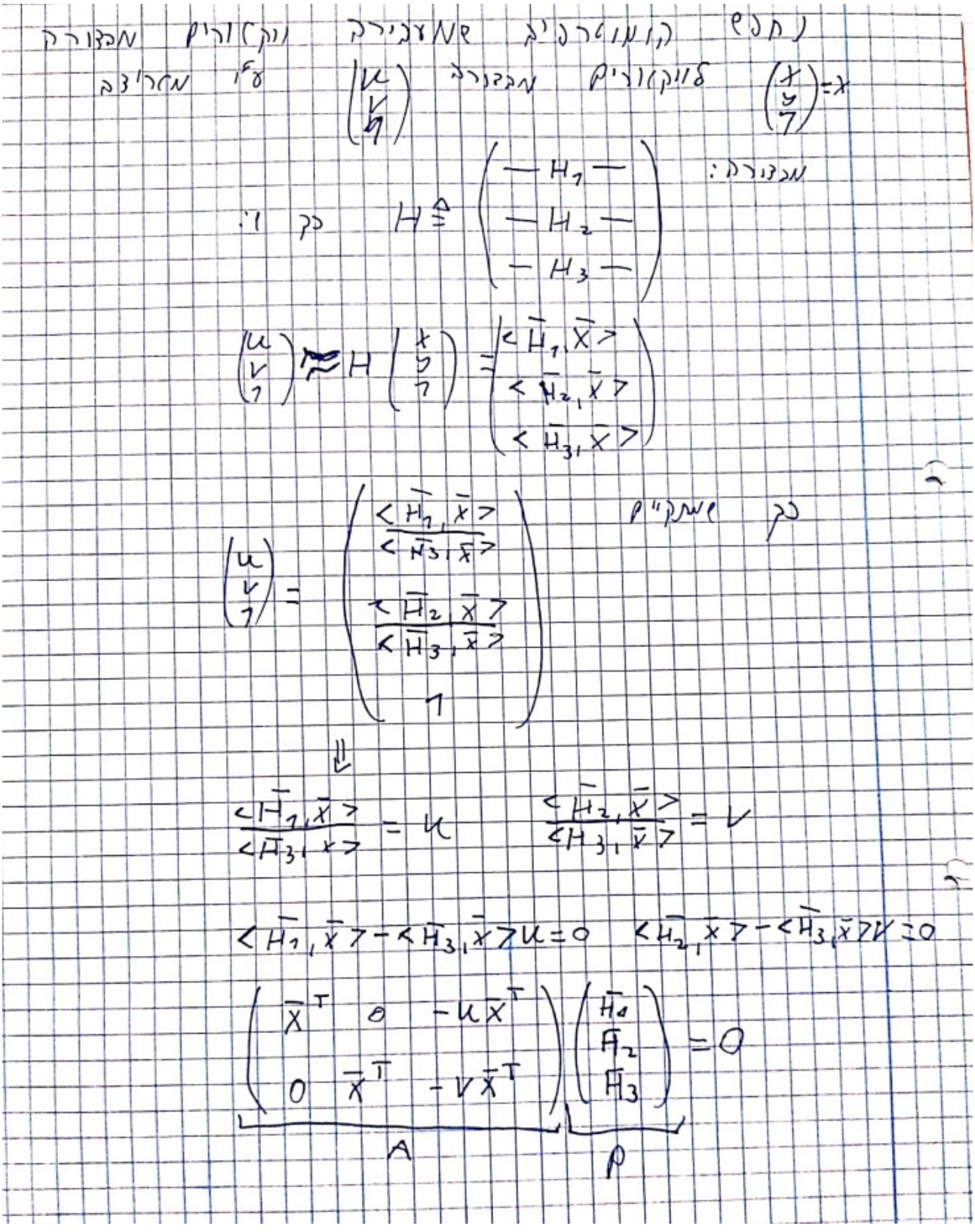
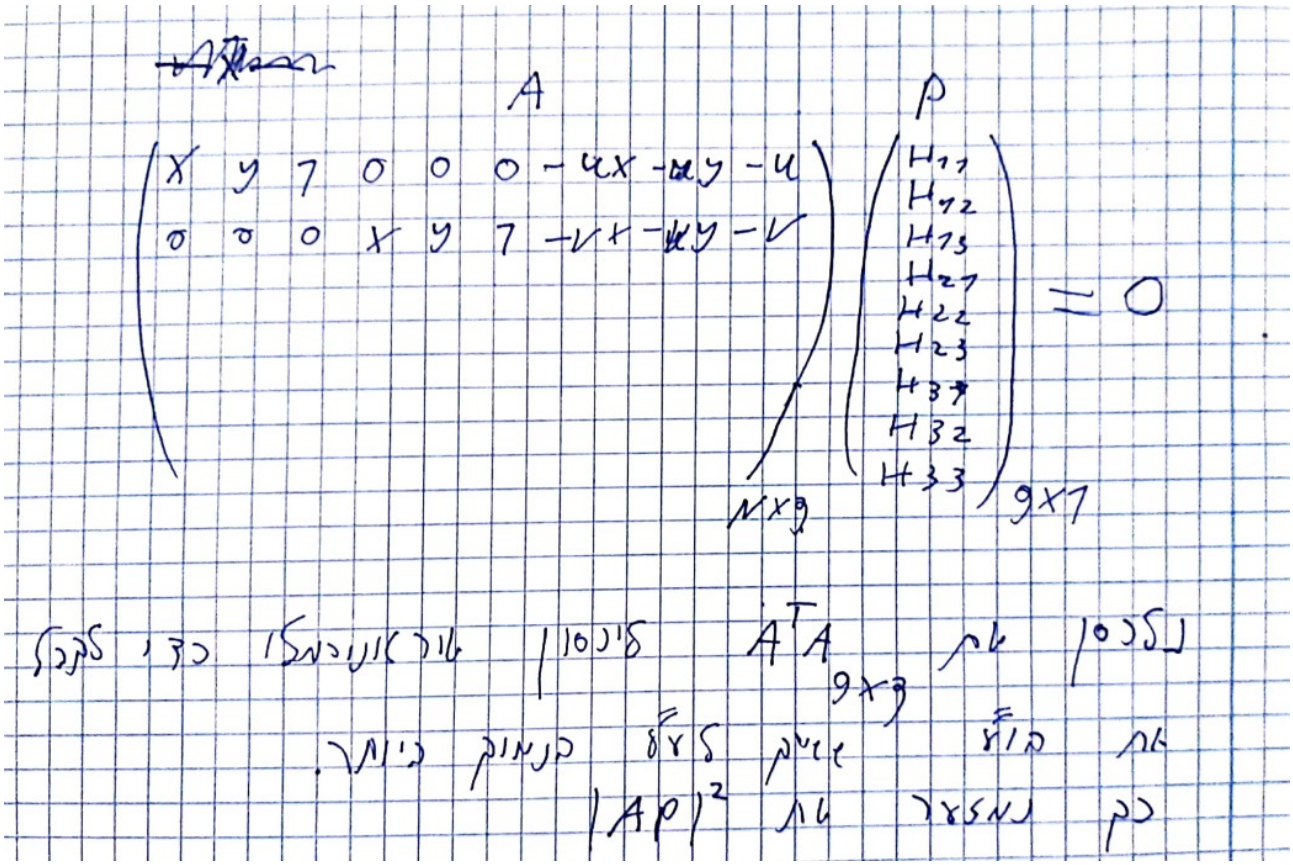
Q1

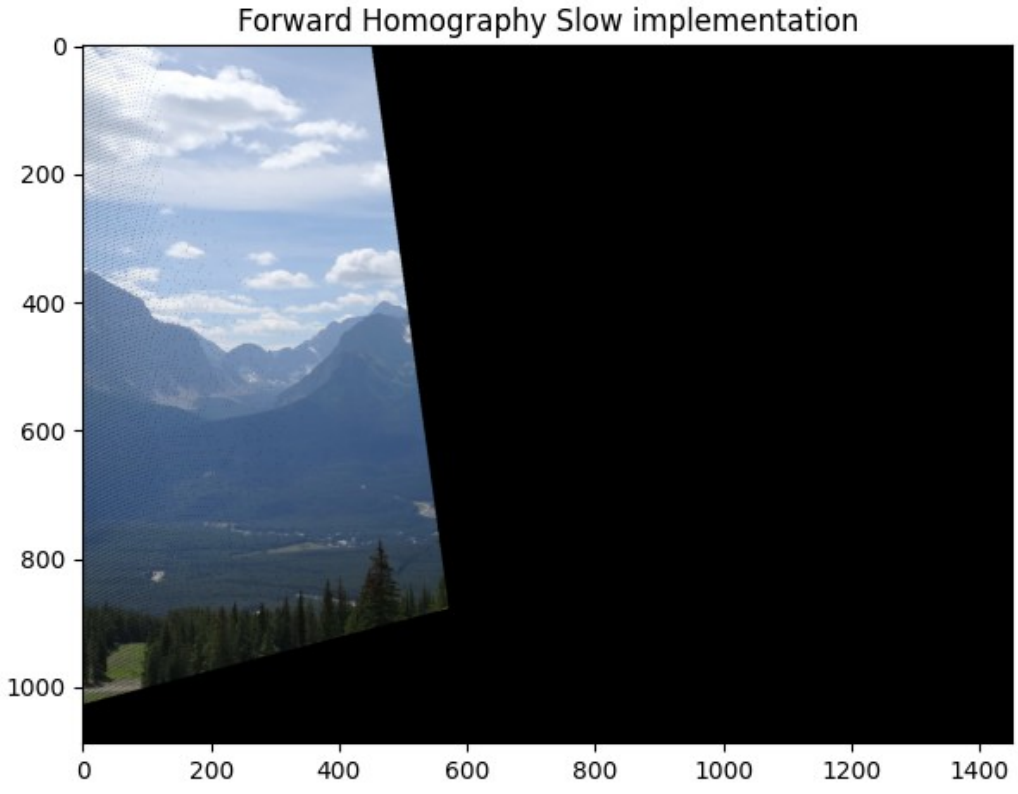




Q3

Naive Homography 0.0004 sec
[[1.12313781e-03 1.64757662e-04 -9.99919585e-01]
 [1.05117245e-05 1.05462483e-03 -1.25622165e-02]
 [2.96940746e-07 4.35706350e-08 7.82907867e-04]]

Q4

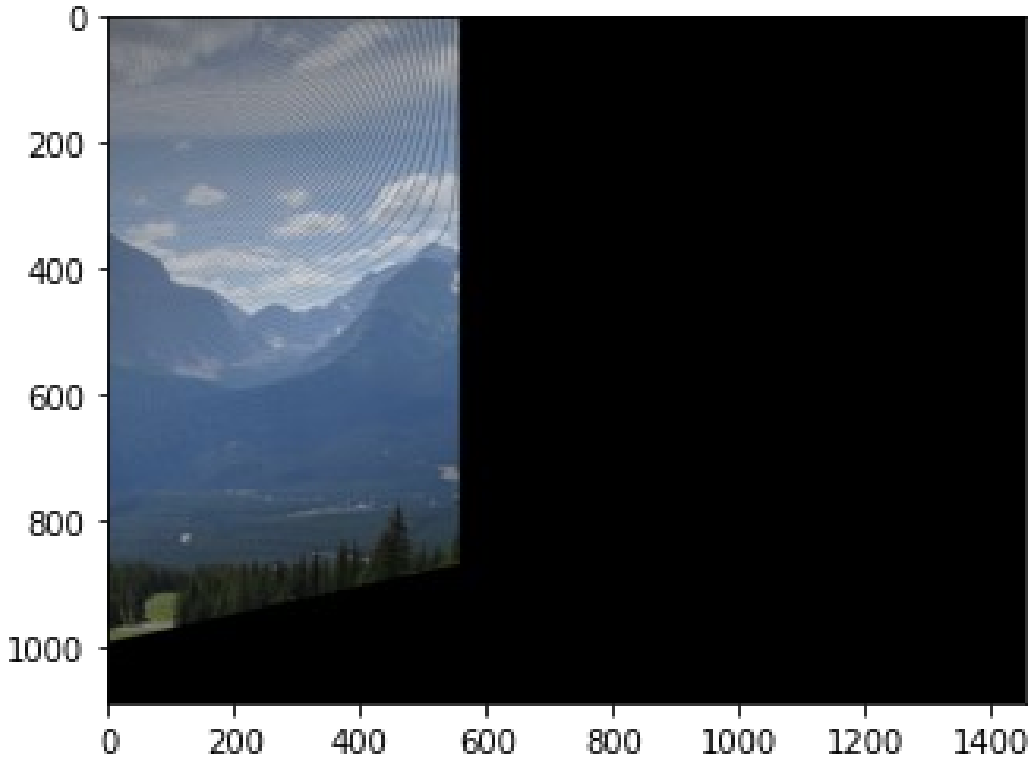


Q6

The two problems are:

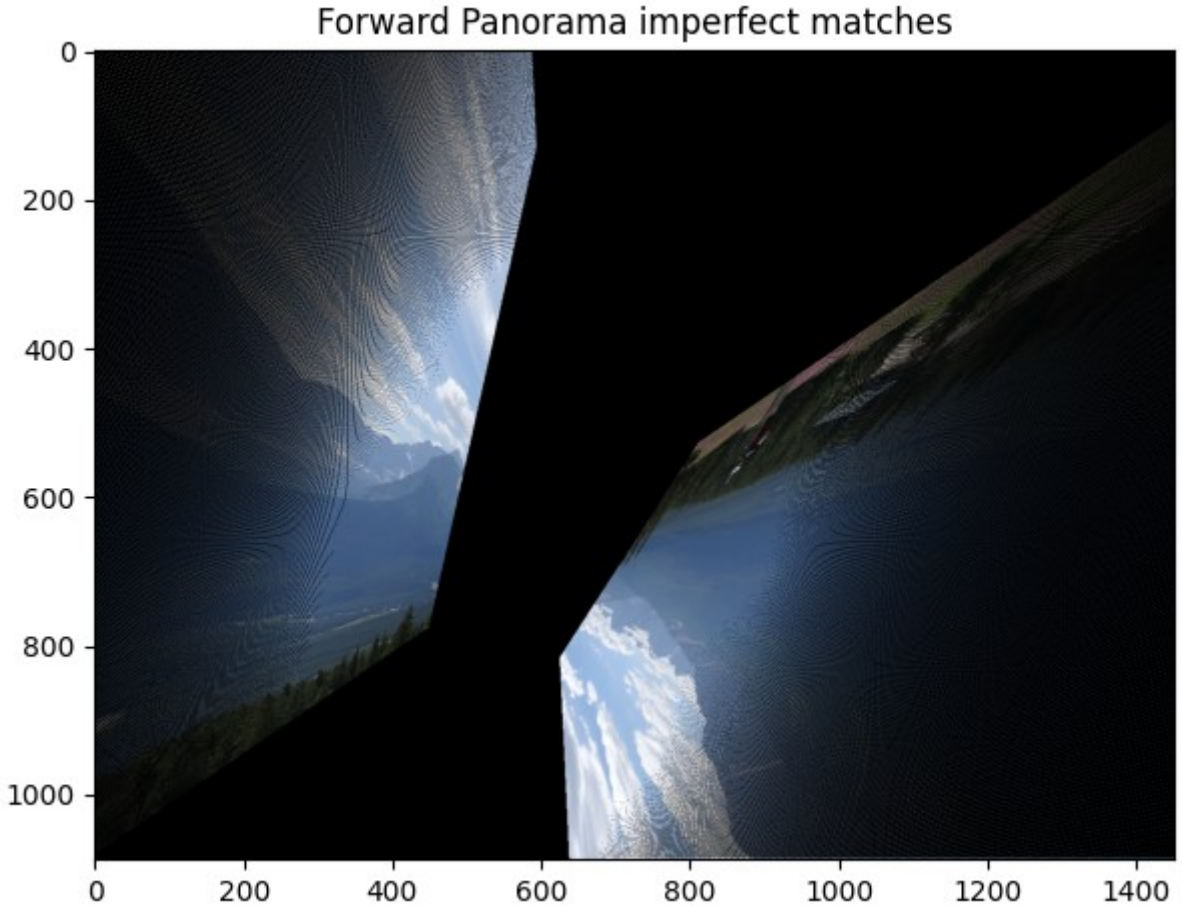
1) The forward mapping doesn't necessarily produce integer fractional pixel, and we have to decide manually where to map it.

2) We are not guaranteed that all the pixels in the destination image have been covered via source pixel that passed through the transform, so we might have "empty" spaces



In this example (from further section) the source image went through the homography and part of it became blurry because of these problems.

Q7



Defiantly using matches instead of perfect matches resulted a bad homogrpahy, thats because not we include outliers points.

Q10

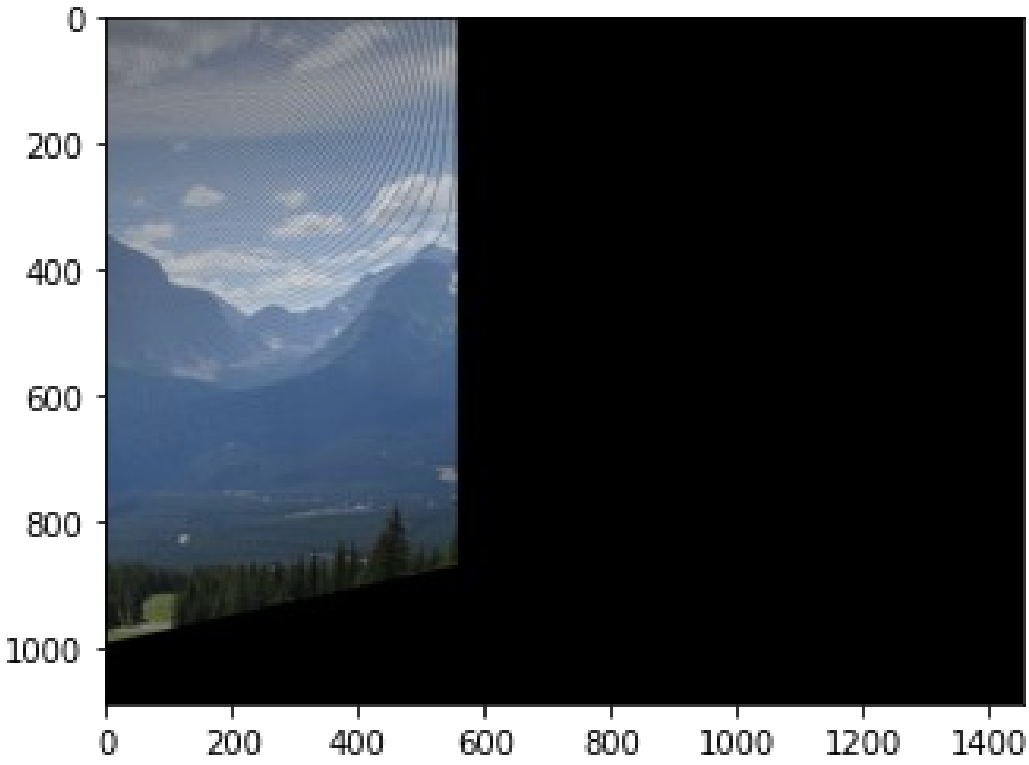
As we have seen in class:
The 4 power its because we use 4 points at each iteration.

$$k = \frac{\log(1-0.9)}{\log(1-0.8^4)} = 5$$

$$k = \frac{\log(1-0.99)}{\log(1-0.8^4)} = 9$$

Naive Homography for imperfect matches 0.0002 sec
[[5.69473545e-04 1.27553946e-04 -6.07820712e-01]
 [6.08182336e-04 4.71576145e-04 -7.94073110e-01]
 [9.21258229e-07 3.74324571e-07 -9.71767385e-04]]
Naive Homography Fast computation for imperfect matches takes 0.2003 sec
Naive Homography Test 0.0002 sec
[0.16, 456.3073545253095]
RANSAC Homography 0.0028 sec
[[1.12681437e-03 1.56712769e-04 -9.99903289e-01]
 [1.70120413e-05 1.04170504e-03 -1.37995286e-02]
 [3.07084708e-07 2.76209271e-08 7.78621733e-04]]
RANSAC Homography Test 0.0001 sec
[0.8, 7.449965919289532]

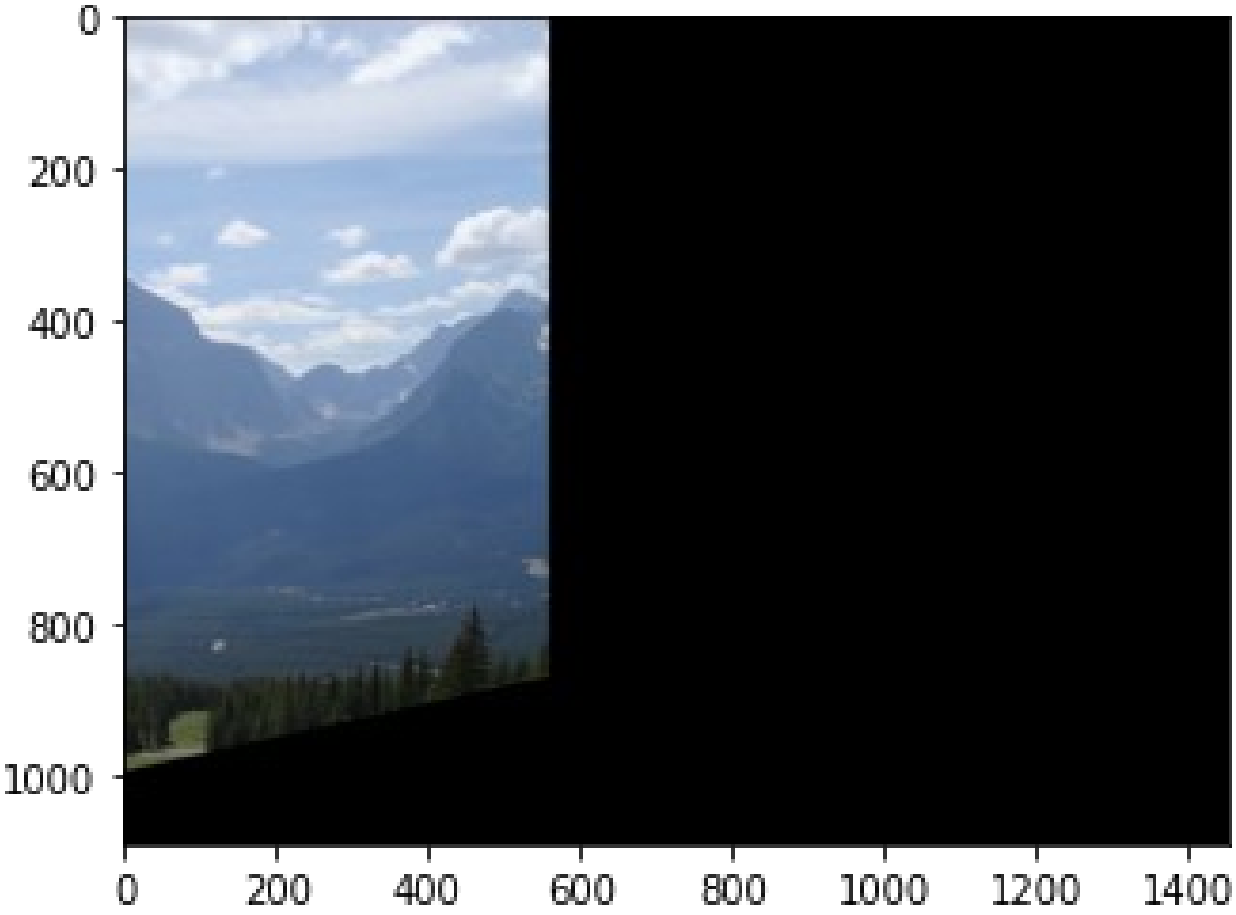
Using ransac we got diffrent homography and image:



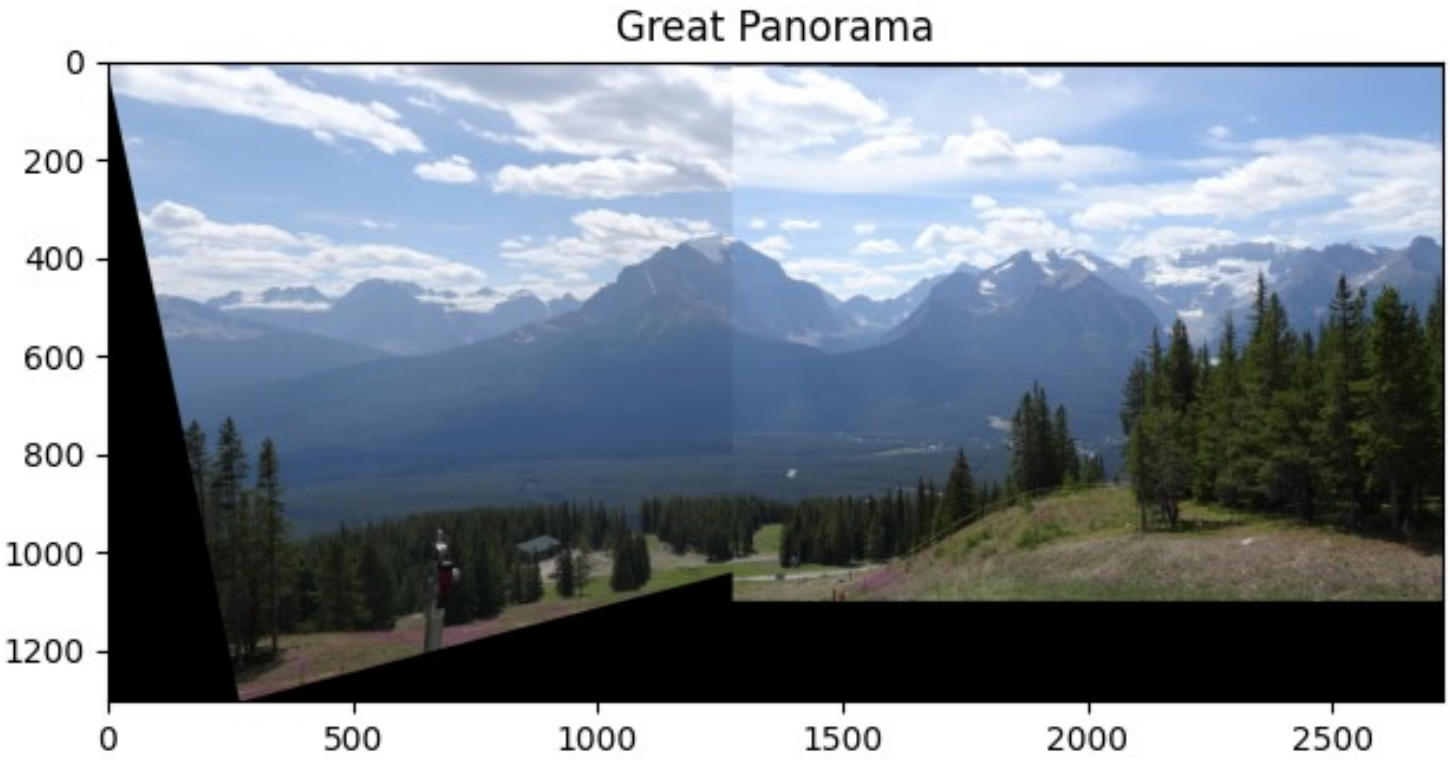
Which seems identical to the one we got using the perfect matches.

Q13

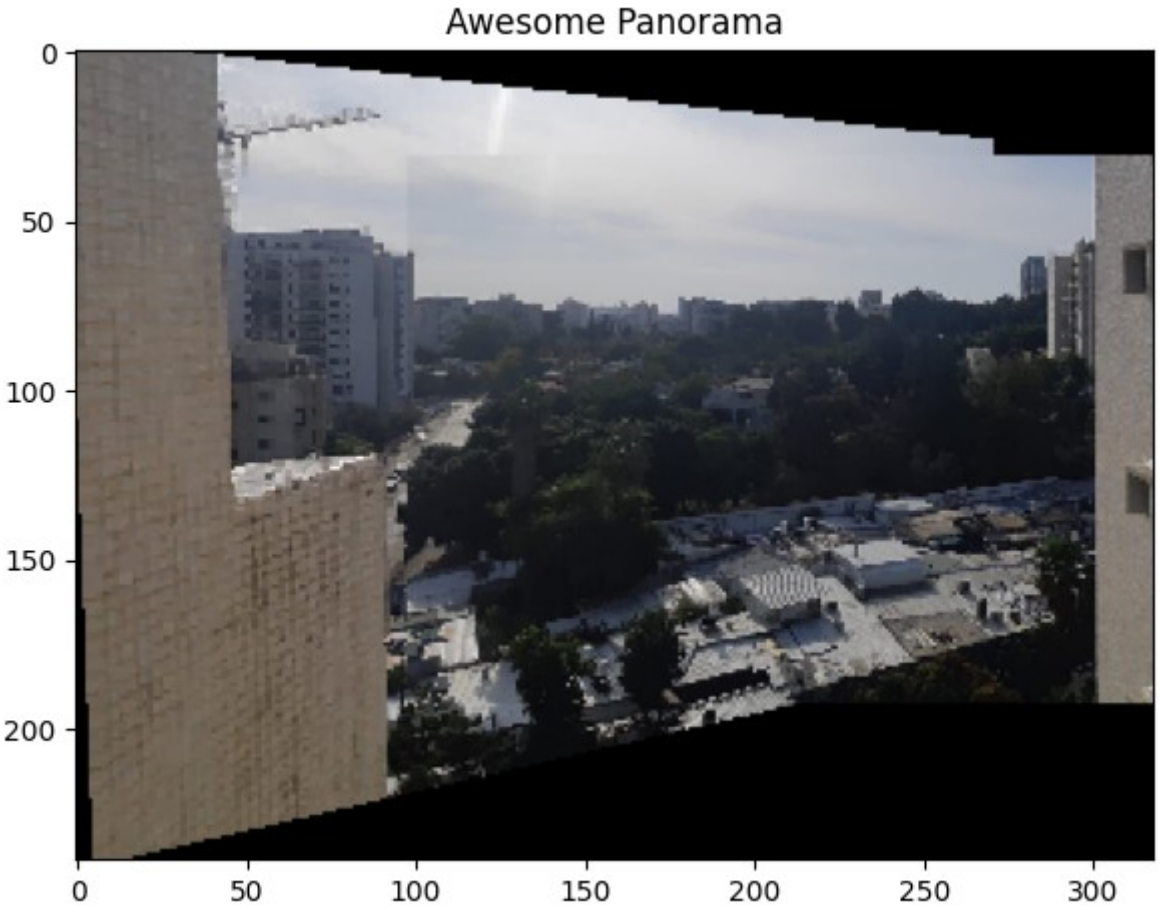
The backward mapping cleared all the blurry parts that we have seen before:



Q16


Great Panorama
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500

Q17


Awesome Panorama
0
50
100
150
200
0
50
100
150
200
250
300