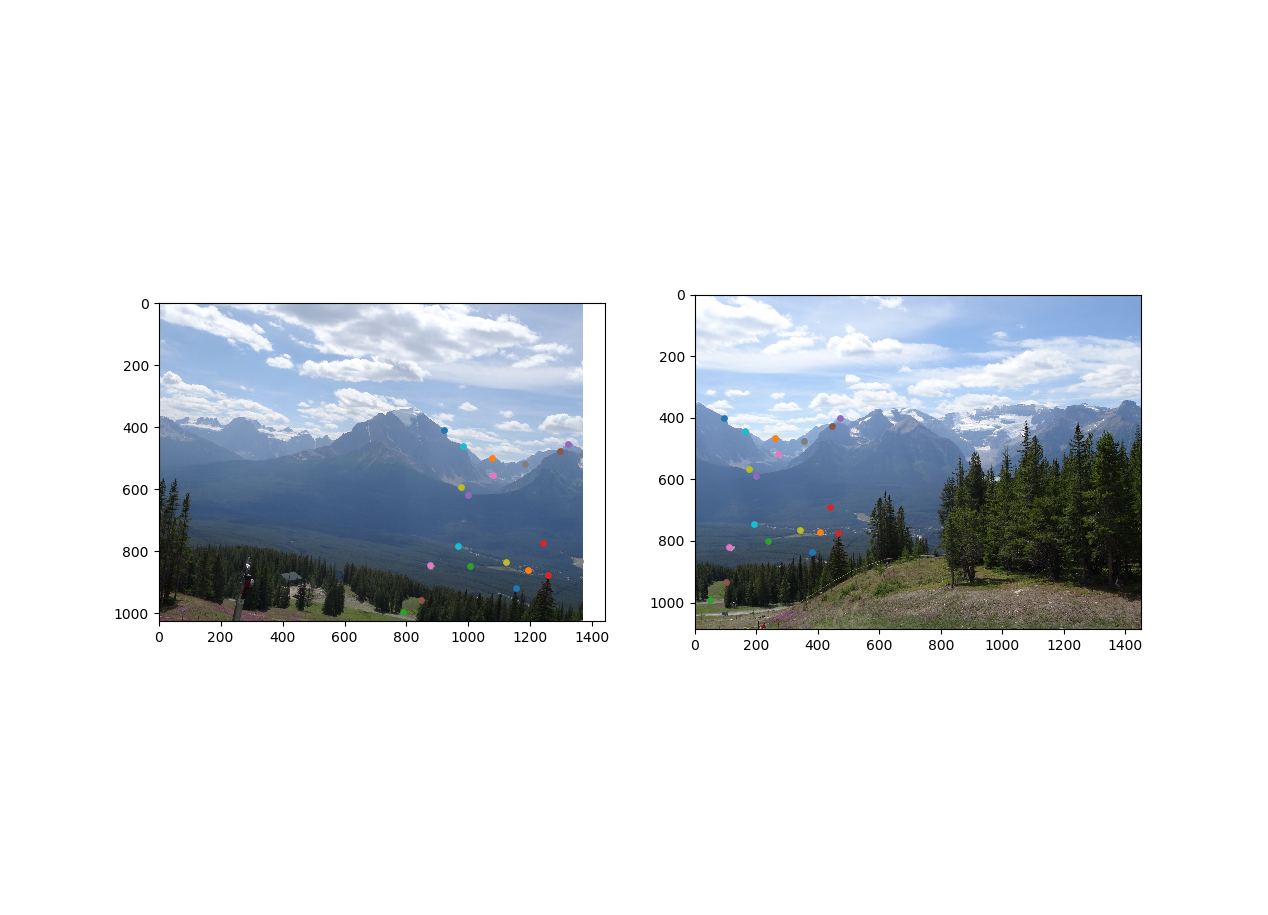
# Computer Vision Ex1 – Solution

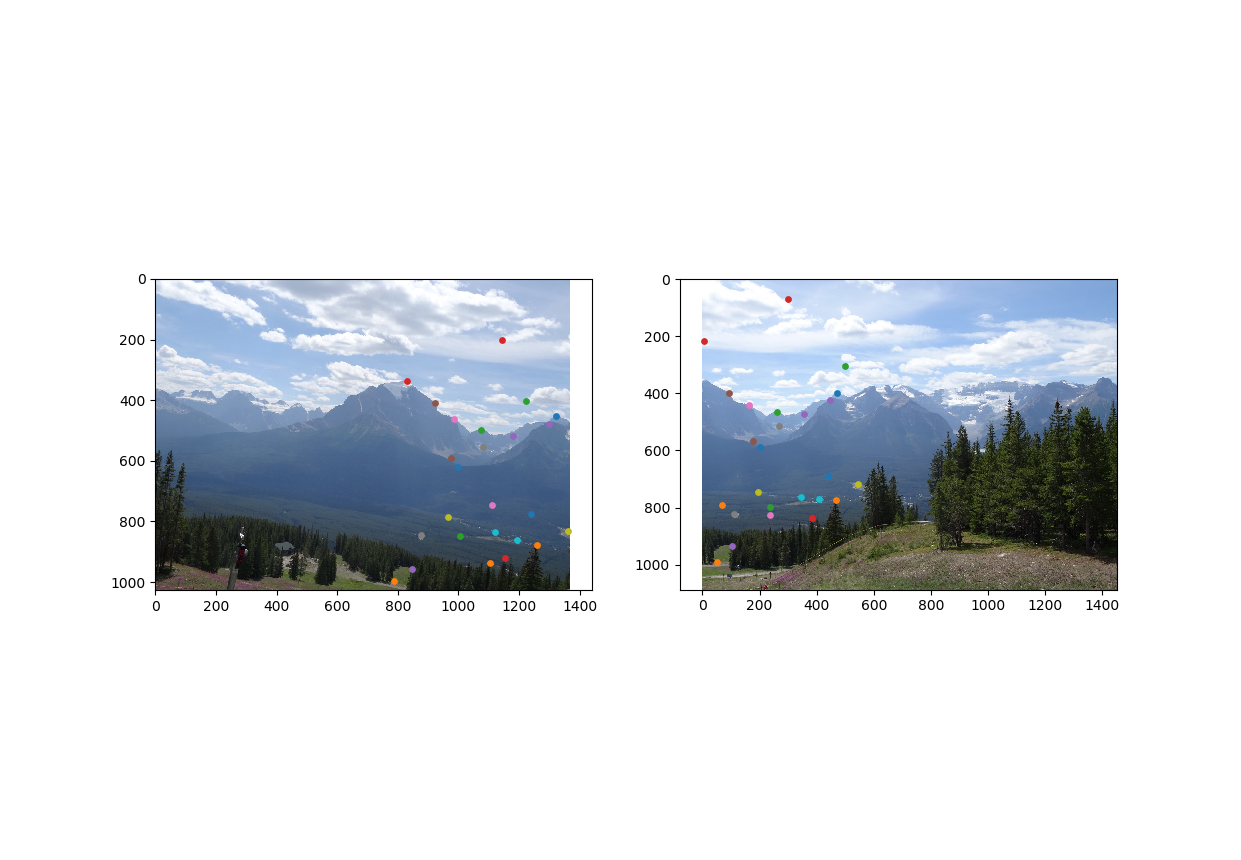
## The Problem

Plotting the given interest points on the images:



Checking the matching point, indeed we see that the points of similar color are marking the same points in both images.

When loading the non-perfect matching points file the following can be seen:



We can see for example points in the clouds that mark places that are different between the two images.

## Part A

1. Homography is defined by:

We learned in class that to solve an homography we need at least 4 pairs of matching points between two images (that are either only rotated or the in which the subject is planar) in order to get the homography. In addition, no linear dependent rows can be present in the constructed matrix.

For each pair of points we get two linear equations:

Where is a col vector representing the point in the src image ; () is the coords of the matching point in the dst image and are the rows of the homography matrix .

Similarly, we can write it in a different way:

Then:

Put in matrix, we get the following eq:

As stated above, each matching pair will add two additional rows to the final matrix A.

To get the conversion matrix from the equation system, we first need to calculate

Then we need to find the eigen vector that corresponds to the smallest eigen value, this can be done using SVD. The eigen vector that we got will be reordered to get the final homography matrix.

3. Output taken from the code after computing the homography using compute\_homography\_naive:

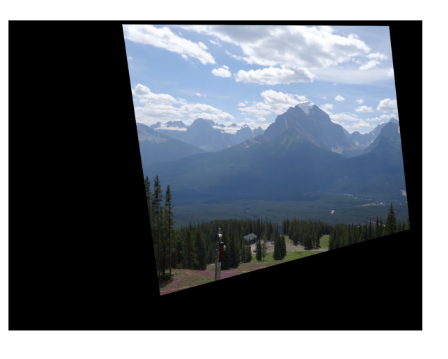
Naive Homography 0.0010 sec

[[ 1.43457214e+00 2.10443232e-01 -1.27718679e+03]

[ 1.34265156e-02 1.34706123e+00 -1.60455878e+01]

[ 3.79279299e-04 5.56523152e-05 1.00000000e+00]]

4. Using the coefficients from the previous section (forward mapping) and applying it to the src image we get the following result:



It is worth mentioning, we translated the image before applying the result, otherwise the resulting image was cropped.

5. There are two problems with forward mapping:

Fractional pixel coordinates - we are not going to necessarily fall on an integer pixel coordinates, most likely not. This poses two problems for forward mapping:

1. Overlaps – there may be two pixels that after being applied with the forward mapping they fall on the same pixel in the output image. The last one we set will determine the pixel value.
2. Possible gaps (holes) in the output – there is no guarantee that we are going to fill all the pixels in the output image since we may have two adjacent pixels that after being applied with the forward mapping will fall in the output image with a gap between them. This will result with gaps in the output.

6. In this section we assume a naïve approach while the data that we load (matches.mat) is “contaminated” with incorrect matching points.

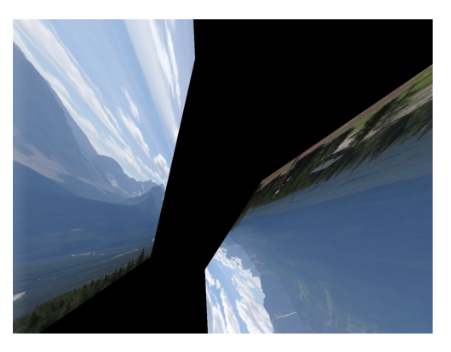
Naive Homography 0.0020 sec

[[-5.86018364e-01 -1.31259750e-01 6.25479626e+02]

[-6.25851766e-01 -4.85276766e-01 8.17143196e+02]

[-9.48023403e-04 -3.85199763e-04 1.00000000e+00]]

The src image after applying the transformation:



We can see that adding the outliers has a significant affect on the resulting panorama. A small amount incorrect matching points yields an unusable result.

Homography assumes that either the subject is planar or there is only rotation, and it cannot represent transformation that does not apply to these rules. Adding the incorrect matching points drastically changes the resulting matrix from its correct state.

9. Suppose there are 30 match points and it is known that 80% of them are correct.

The formula we developed in class is the following:

Where

* p – RANSAC success probability that we want (90%, 99%)
* k – number of iterations required to achieve RANSAC success probability of p
* w – probability to pick a correct matching (0.8) probability to choose a correct matching
* n – number of minimal matching needed to calculate an homography (4)

So we have for **90% confidence**:

And for **99% confidence:**

10. Using RANSAC with the method compute\_homography, the following homography transformation is the result:

RANSAC Homography 0.0030 sec

[[ 1.43457214e+00 2.10443232e-01 -1.27718679e+03]

[ 1.34265156e-02 1.34706123e+00 -1.60455878e+01]

[ 3.79279299e-04 5.56523152e-05 1.00000000e+00]]

This is actually precisely the same as the transformation we got with the naïve approach and the perfect matching points even though now we have outliers in the data. Since the algorithm found all the original inliers (from the perfect data and does so with high probability) and calculated the transformation using all the inliers, consequently, we got the same result.

**After mapping the src image using the coefficients above, we got the following result:**



We can see that in this section we got an image that is similar or even identical to the one we got after mapping using the transformation, we got with the perfect matching points. This tells us that RANSAC worked very well and ignored the outliers from the “contaminated” matching points file.

Compared to section 6, there is no question in the effectiveness and significance of RANSAC. From an unusable transformation we got a transformation that is identical (with high probability) to the one calculated on the perfect data.

11. We compare our results with and without bi-linear interpolation when applying the homography transformation to the src image.

At first glance there seem to be no difference, but when we zoom in we can spot that the bilinear transformation presents some somoothness effect to the image, it is visible mostly on high contrast edges and in the margins of the image. We show closeups below.

|  |  |
| --- | --- |
| Without bilinear interpolation | Without bilinear interpolation |
|  |  |

margins are smoothed since they are interpolated with black pixel values when outside the image.

|  |  |
| --- | --- |
| Without bilinear interpolation | Without bilinear interpolation |
|  |  |

High contrast edges show a bit more smoothness.

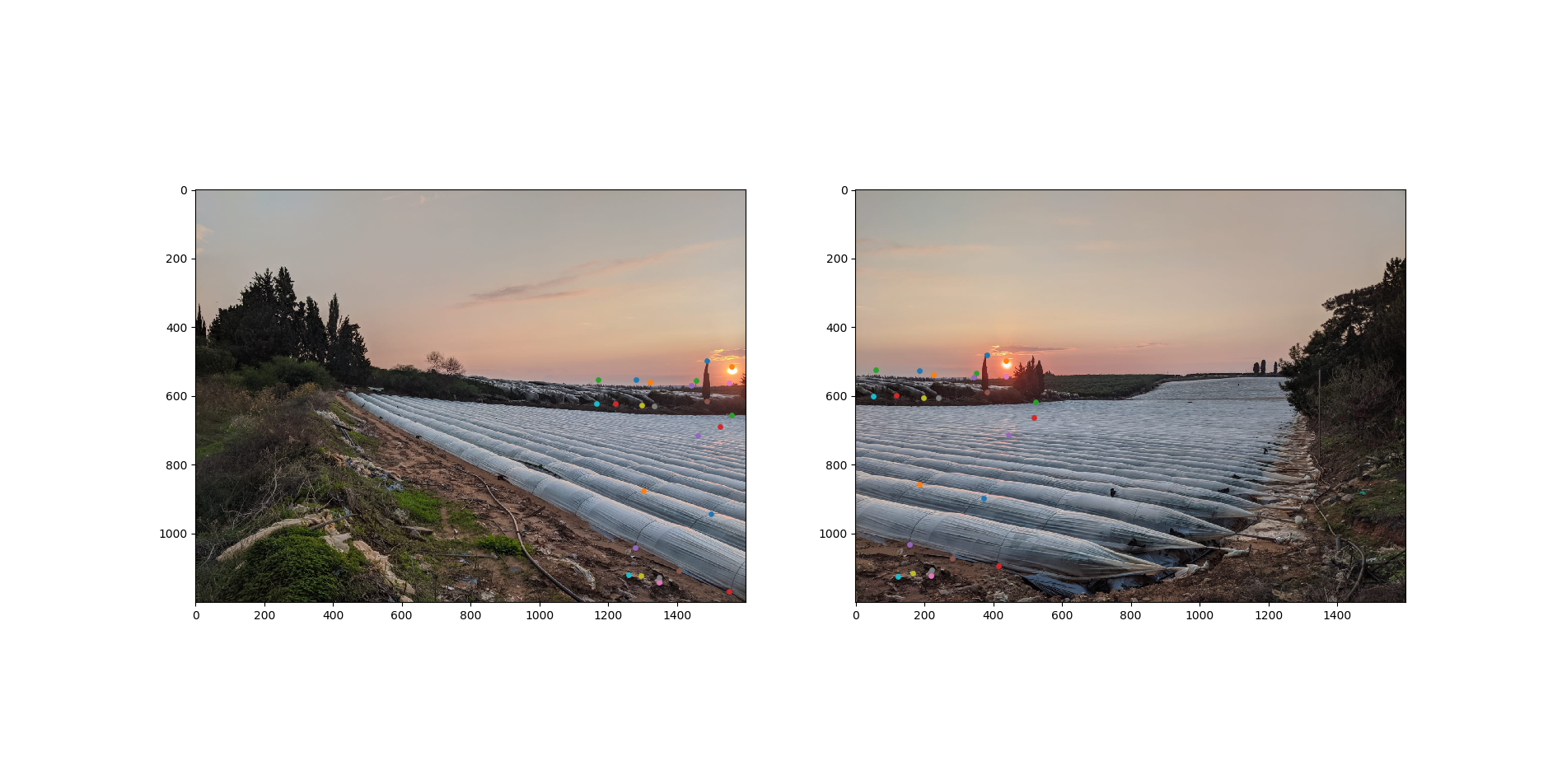
13. The great panorama image



14. Our own panorama images, taken in Kadima-Tzoran

* src\_test  
  
* dst\_test  
  

Our marked points, including the outliers, most of the outliers can be seen near the center of right image:

  
Our final panorama result: