**Assignment 1:**

**Homography & Panorama**

1. While an affine transformation is used for scaling, skewing and rotation, a projective transformation is used to express how the perceived objects coordinates change when the view point of the observer changes.

A projective transformation of coordinates (x,y) in one vector space to (x’,y’) on another vector space, can be expressed by the following equation:

Where H is a 3x3 matrix, and the sign represents an equality up to scale.

Let’s denote coordinates, and their projections with a subindex i.

Thus, we can write the following equations for each pair:

Where holds a degree of freedom since we only demand equality up to scale. Therefore, each pair (and their projection) of coordinates only sum up to 2 constraints on the values of H which can be expressed with the following equations:

Suppose we have n pairs of coordinates. Then we can express the equations as follows:

We denote this matrix as A – a matrix of size 2n x 9, and get a matrix of the form where the solution for x, describe H.

Since the source equality is up to scale, there are actually 8 degrees of freedom for H. Thus only 4 pairs of coordinates are sufficient to find H (We assume that A is not defective).

If n is bigger than 4, equality may not be possible (due to some error in the projective input). In such case, we strive to minimize .

One can notice that:

We denote – a PSD matrix of size 9x9. Thus, our objective is to find x such that is minimized.

Assuming M has 9 linearly independent eigenvectors, we can define – a matrix whose columns are the linearly independent eigenvectors of M, and – a diagonal matrix whose diagonal elements are the corresponding eigenvalues of M.

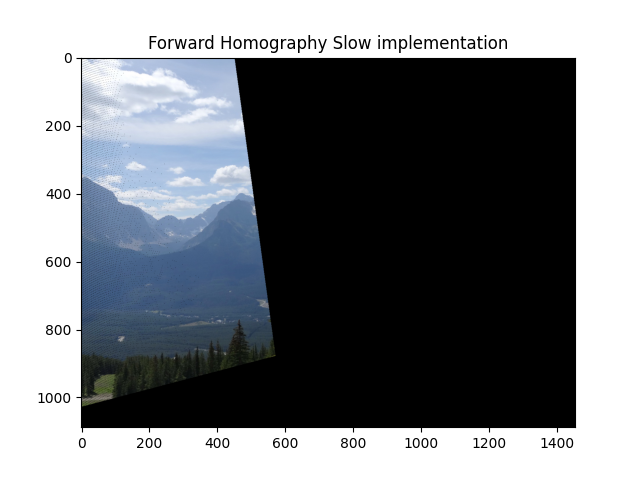
So:

Each vector x can be described as a weighted sum of the eigenvectors of M. From here, it is trivial to see that the desired x that minimizes is the eigenvector corresponding to the smallest eigenvalue of M (which is necessarily not negative since M is a PSD matrix).

1. The function compute\_homography\_naive is implemented and can be viewed in the python src files.
2. After loading matches.mat and using the function implemented under section 2, we found the following matrix:

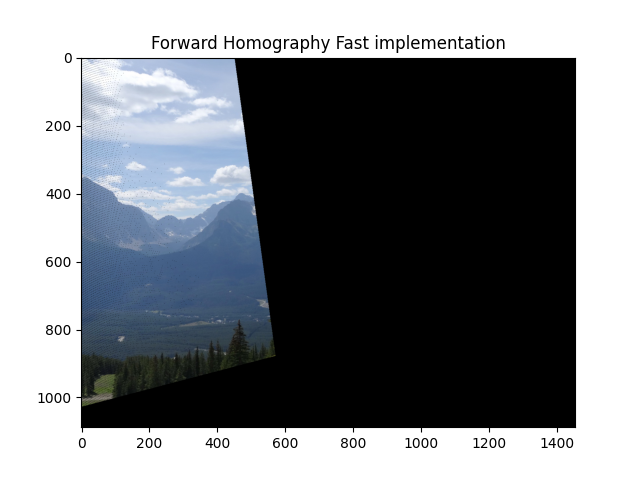
We remember that H is defined up to scale. Thus, a common approach is to represent H where . In such case:

1. Below is the image generated using compute\_forward\_homography\_slow function:



It is clear to see that the projected image is only the part of the src image which is in the coordinates domain of the target image.

1. Below is the image generated using compute\_forward\_homography\_fast function:

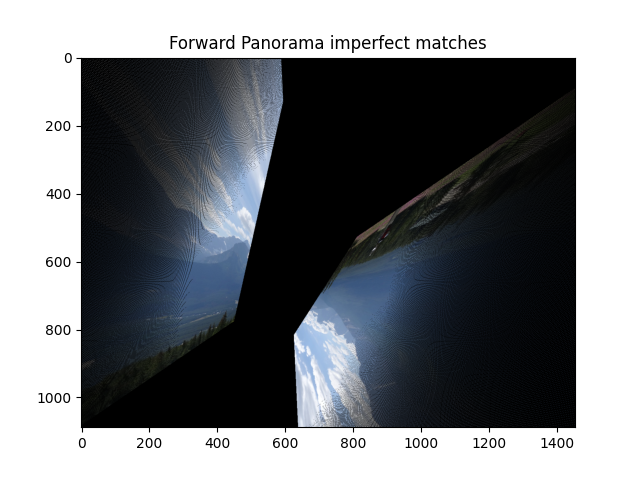


The output appears to be the same as of section 4, as expected.

1. Forward mapping has 2 main problems:

* Upon projecting a coordinate to the destination image coordinate space, the pixel coordinate may not necessarily be an integer, but rather in between the coordinates in the destination space of pixels. In such case, the forward mapping rounds up the coordinate location. This brings some distortion in the projection.
* The added process of rounding up the coordinate location + the basic method the projection is done by, yields a situation where some coordinates in the destination image do not have a source coordinate which was projected to it. This yields in black spaces in between projection (As can be viewed in the images at section 4, 5).
* The additional step of rounding up the coordinate location means that some coordinates are mapped to the same location in the destination image. Thus, the later projection overrides the first one.

1. Below is the image generated using compute\_forward\_homography\_fast function, where we used the imperfect matches instead of the perfect ones:



As we can see the addition of imperfect matches (outliers), yields in a full corruption of the homography matrix found. The problems of section 6 still appear (However it is not the main problem of the transformation now).