Homework Assignment 1

Mai Dabas (315026294) & Adi Green (313472417)

Computer Vision

Part A: Homography computation

1. As seen in the lecture, the system of equations for **projective** transformation is:

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \cong \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Each point (u_i, v_i) in the source coordinate system matches a point (u'_i, v'_i) in the destination system.

To get the conversion matrix, we build the following equation system,

where: $x' = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}, x = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$ $and\ H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix}$

$$x' = \vec{H}x$$

$$\Rightarrow \frac{u'}{1} = \frac{H_1 x}{H_3 x} \Rightarrow u'\vec{H}_3 x - H_1 x = 0 \Rightarrow (u'H_3 - H_1)x = 0$$

$$\Rightarrow \frac{v'}{1} = \frac{H_2 x}{H_3 x} \Rightarrow v'H_3 x - H_2 x = 0 \Rightarrow (v'H_3 - H_2)x = 0$$

From these two equations, we can build this equation:

$$\begin{pmatrix} x^T & 0 & -ux^T \\ 0 & x^T & -v'^{x^T} \end{pmatrix} \begin{pmatrix} H_1^T \\ H_2^T \\ H_3^T \end{pmatrix} = 0$$

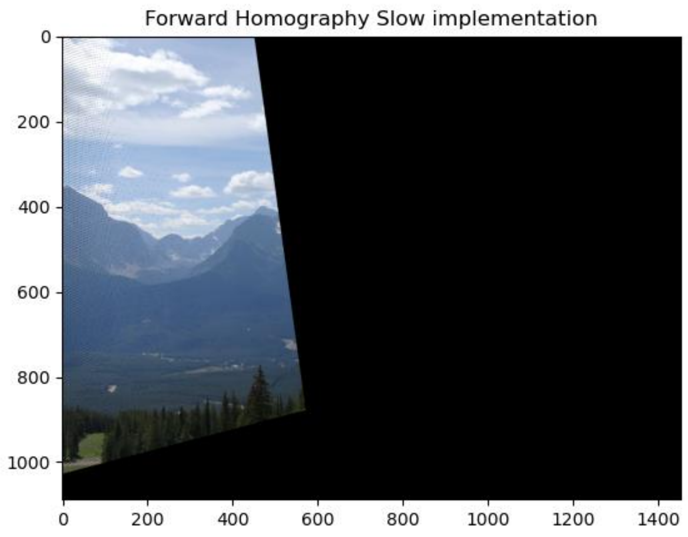
And find H such that AH=0, while adding an extra constraint |H|=1 to avoid the obvious solution of H being all zeros. To estimate H, we need at least 4 points (eight equations from the four points, and since H is up to scale, the last equation is a normalization constraint). If there are more than 4 points (as in our case), we can just keep plugging them as new rows in matrix A (to new rows per point). Then we can use SVD ($A = USV^T$), and select the smallest singular vector of V^T as the solution to H. To get the matrix form of 3x3 we reshape this vector.

2. In the code.
3. The results:

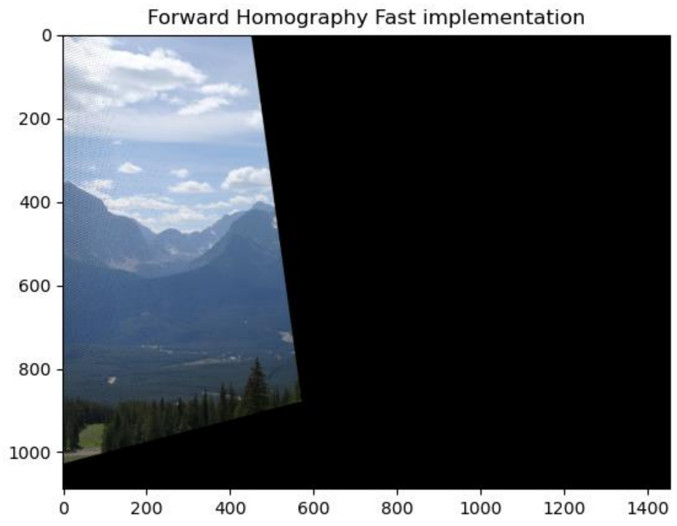
$$H = \begin{bmatrix} 1.12313781 * 10^{-3} & 1.64757662 * 10^{-4} & -9.99919585 * 10^{-1} \\ 1.05117245 * 10^{-5} & 1.05462483 * 10^{-3} & -1.25622165 * 10^{-2} \\ 2.96940746 * 10^{-7} & 4.35706349 * 10^{-8} & 7.82907867 * 10^{-4} \end{bmatrix}$$

Part A2: Forward mapping slow and fast:

4. In the code.



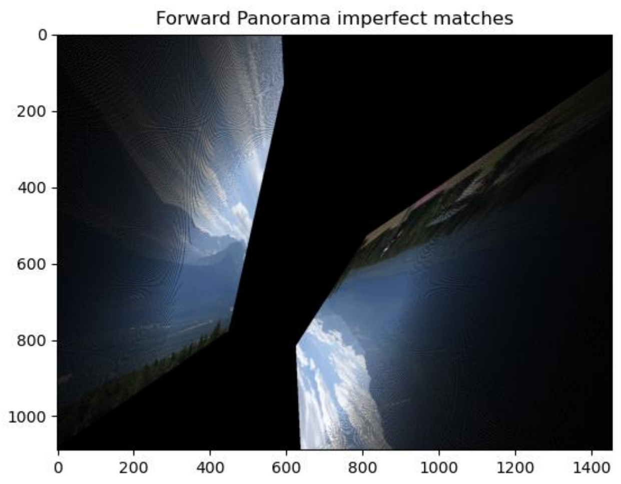
5. In the code.



6. The forward mapping suffers from 2 issues:
   a. Integer coordinates can be mapped into non-integer coordinates.
   b. Leaving black pixels in the target as there is no guarantee that the mapping will fill all pixels in the target image.

In our images, we rounded the pixels to deal with problem a, but you can see in the images above black dots in some areas of the image which is cause by problem b.

7. The resulting image:



In this case, the image is not as we expected. Due to some mismatching points the result is distorted. In (3) we calculated the homography matrix when the matching points are perfectly matched between the source and the destination images, and we can see that in this case, where the matching points include outliers, the homography matrix is also different.

$$H = \begin{bmatrix} -5.69473545 * 10^{-4} & -1.27553946 * 10^{-4} & 6.07820712 * 10^{-1} \\ -6.08182336 * 10^{-4} & -4.71576145 * 10^{-4} & 7.94073110 * 10^{-1} \\ -9.21258229 * 10^{-7} & -3.74324571 * 10^{-7} & 9.71767385 * 10^{-4} \end{bmatrix}$$

Part B: Dealing with outliers

8. In the code.
9. In the code.
10. In class, we saw the following formula to find k – no. of iterations required to fit the model:

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

when:

$w - \ fraction\ of\ inliers = 80\%$

$n - min. number\ of\ points\ required\ to\ fit\ the\ model = 4$

$a)\ \ p - target\ success\ rate = 90\ \%$

$b)\ \ p - target\ success\ rate = 99\ \%$

As for a) $p = 90\%$:

$$k = \frac{\log(1-0.9)}{\log(1-0.8^4)} = 4.37$$

Meaning we need ~5 iterations (rounding up to avoid getting less than the desired success rate).

As for b) $p = 99\%$:

$$k = \frac{\log(1-0.99)}{\log(1-0.8^4)} = 8.74$$

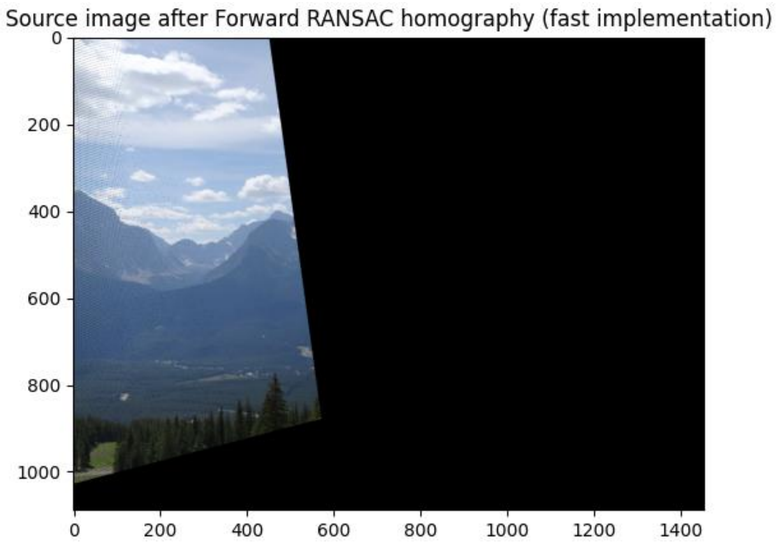
Meaning we need ~9 iterations (rounding up to avoid getting less than the desired success rate).

In order to cover all options, given 30 match points, and assuming we don't choose the same group of four points more than once: $k = \binom{30}{4} = 27{,}405$ iterations are needed.

11. In the code.
12. RANSAC homography:

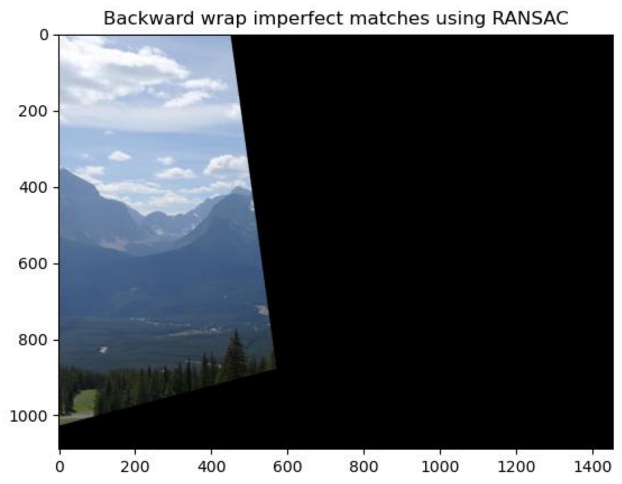
$$H = \begin{bmatrix} -1.12313781 * 10^{-3} & -1.64757662 * 10^{-4} & 9.99919585 * 10^{-1} \\ -1.05117245 * 10^{-5} & -1.05462483 * 10^{-3} & 1.25622165 * 10^{-2} \\ -2.96940746 * 10^{-7} & 4.35706349 * 10^{-8} & -7.82907867 * 10^{-4} \end{bmatrix}$$

The source image after projective transform using forward mapping (fast implementation):



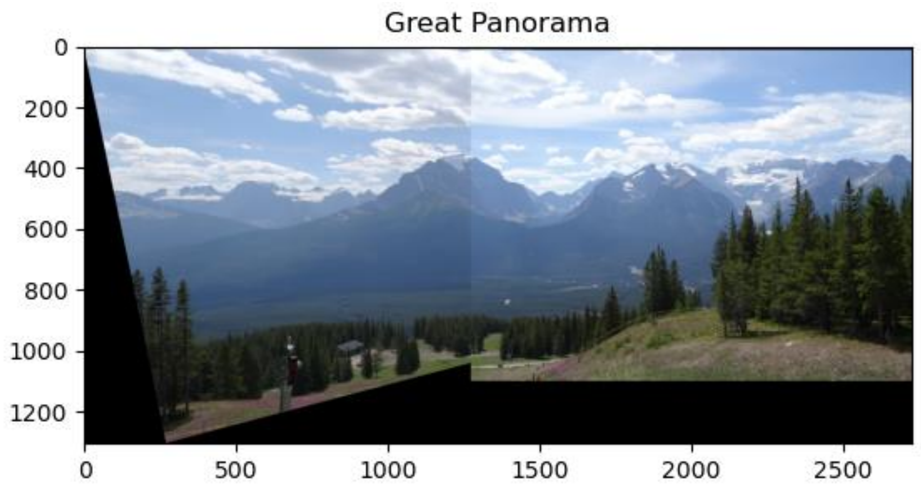
The transformation coefficients homography matrix is identical (with negative sign) to the one presented in section 3 where we used perfect match points, so the purpose of the RANSAC algorithm to remove outliers and gain a cleaner result has worked. The image is also identical to the image gained when using the perfect matches (section 5) as opposed to the image gained when there were outliers in the match points (Section 7).

13.

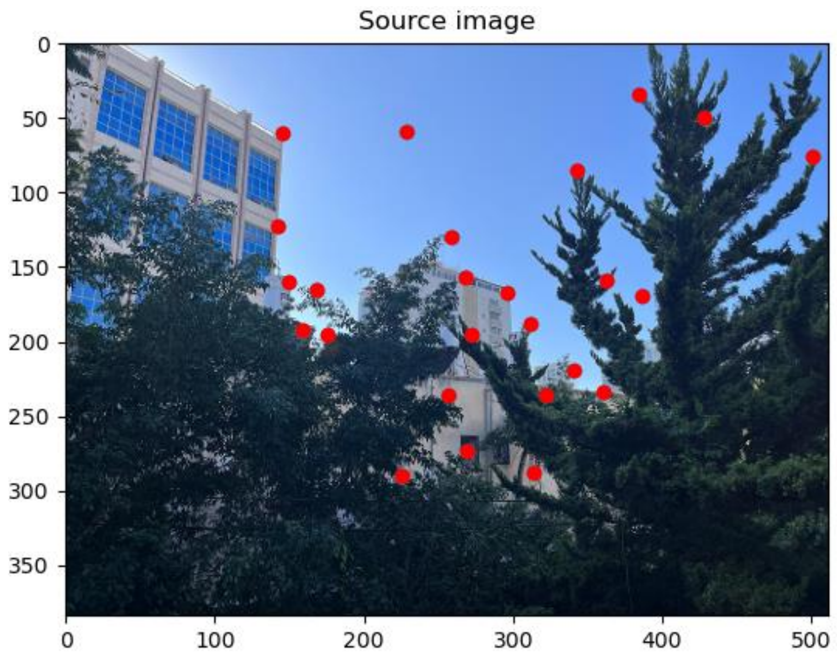


Using interpolation and backward wrap has filled the black “holes” that we can see in section 12 – which is one of the main issues that forward mapping suffers from.

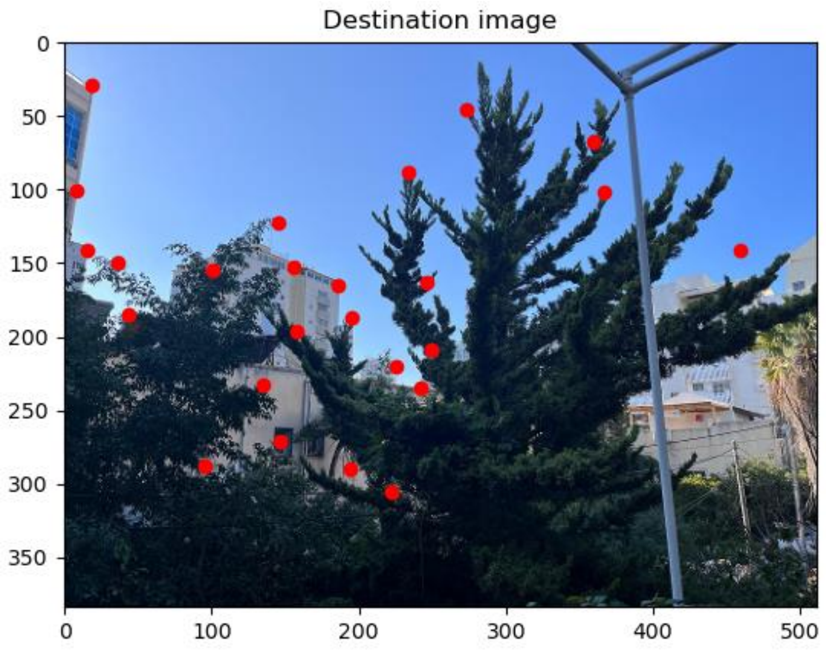
14. In the code.
15. In the code.
16. The output panorama image of the ‘panorama’ function for src.jpg and dst.jpg images, setting 80% inliers and max. error of 25 pixels:



17. Our own source image (with marked match points):

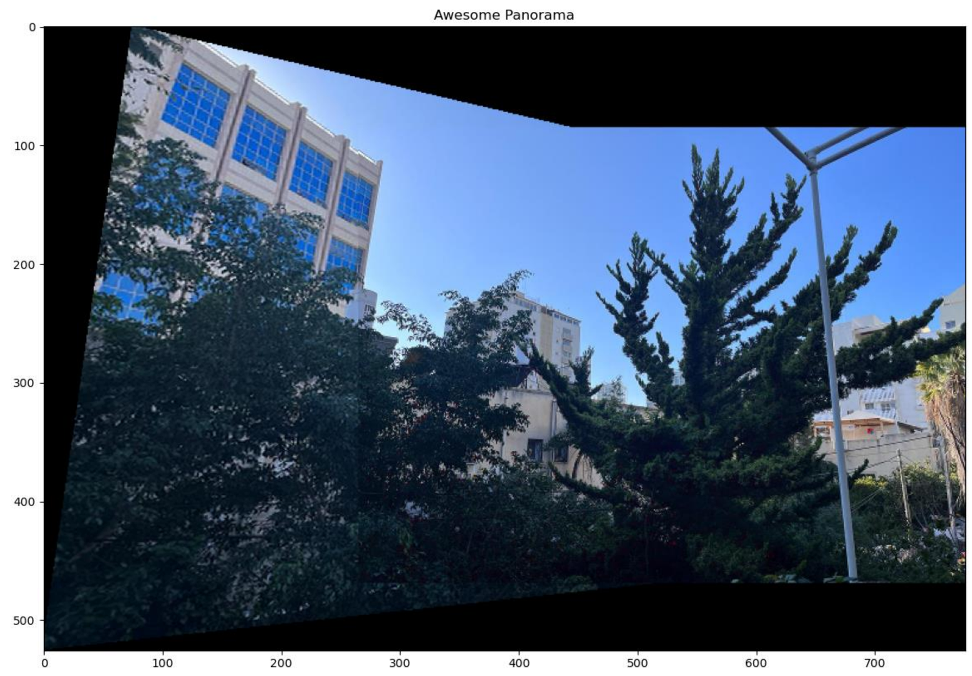


Our own destination image (with marked match points):



The marked match points in the source and the destination images consist of 4 mismatched points.

The output panorama image of our own images:



Reversed panorama image: