TEL AVIV UNIVERSITY

Computer Vision

2022-2023

Assignment #1

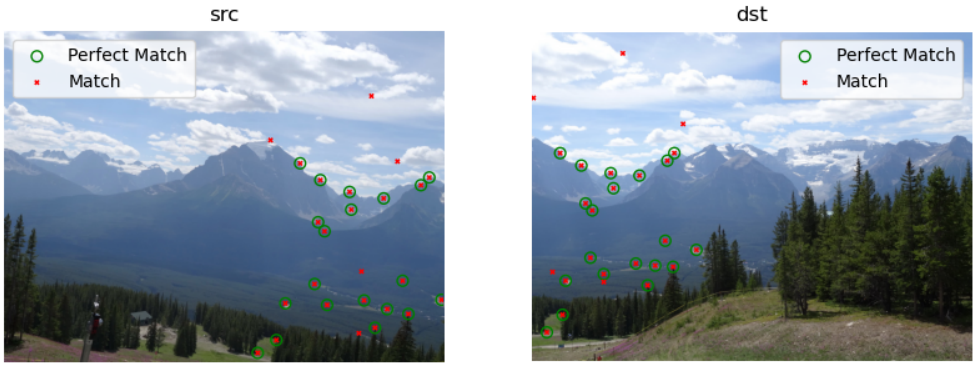
Students

ID1: 301613501

ID2: 305207946

Preparatory Steps

In the images below we can see the matched points vs the perfect match points on the source and destination images.



Part A1: Homography Computation

1. Given a Cartesian coordinate point $p_w = (x_w, y_w, z_w)$ from the 3D space of the world, it is possible to represent the point in Homogeneous coordinate $\tilde{p}_w = (\tilde{x}_w, \tilde{y}_w, \tilde{z}_w)$ using the following transformation:

$$\begin{pmatrix} \tilde{x}_w \\ \tilde{y}_w \\ \tilde{z}_w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

This transformation can be also denoted by the following equations

$$\tilde{x}_w = f x_w; \quad \tilde{y}_w = f y_w; \quad \tilde{z}_w = z_w \qquad for\ non-zero\ scalar, f$$

This point can be projected onto image plain that received by a pinhole camera according to the following transformation that we developed in class

$$p = (x, y) \quad \rightarrow \quad \tilde{p} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix}$$

This point can be projected onto a new image plain (destination image plain) by a projective transformation. Such that

$$\tilde{P}_{dst} = \overline{H}\tilde{P}_{src}$$

Where $\overline{H}$ is the Homography matrix that given by:

$$\overline{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

Since we are in Homogeneous coordinate, we can constraint dividing H by h_{33} so we will get the following:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix}$$

This matrix is taking into account the intrinsic and extrinsic parameters of the camera, real world constrains (i.e., where the camera is located in the world and its orientation) and the fact that image plain doesn't have z-element, so we can transform the image ("**source image**") to another image plain ("**destination image**") by the Homography transformation ("**projective transformation**").

This transformation gives us the pixel in the destination image, but in Homogenous coordinate.

$$\tilde{u} = h_{11}\tilde{x} + h_{12}\tilde{y} + h_{13}$$

$$\tilde{v} = h_{21}\tilde{x} + h_{22}\tilde{y} + h_{23}$$

$$\tilde{w} = h_{31}\tilde{x} + h_{32}\tilde{y} + 1$$

Hence, we should divide each coordinate in $\tilde{w}$, finally we will get the pixel in the destination image in Cartesian coordinate by:

$$u = \frac{\tilde{u}}{\tilde{w}} = \frac{h_{11}\tilde{x} + h_{12}\tilde{y} + h_{13}}{h_{31}\tilde{x} + h_{32}\tilde{y} + 1}$$

$$v = \frac{\tilde{v}}{\tilde{w}} = \frac{h_{21}\tilde{x} + h_{22}\tilde{y} + h_{23}}{h_{31}\tilde{x} + h_{32}\tilde{y} + 1}$$

$$w = \frac{\tilde{w}}{\tilde{w}} = 1$$

Let us develop the linear equation matrix system for u

$$(h_{31}\tilde{x} + h_{32}\tilde{y} + 1)u = h_{11}\tilde{x} + h_{12}\tilde{y} + h_{13}$$

$$h_{11}\tilde{x} + h_{12}\tilde{y} + h_{13} - uh_{31}\tilde{x} - uh_{32}\tilde{y} - u = 0$$

$$\vec{h} \triangleq (h_{11} \quad h_{12} \quad h_{13} \quad h_{21} \quad h_{22} \quad h_{23} \quad h_{31} \quad h_{32} \quad 1)^T$$

$$\overrightarrow{a_u} \triangleq (\tilde{x} \quad \tilde{y} \quad 1 \quad 0 \quad 0 \quad 0 \quad -u\tilde{x} \quad -u\tilde{y} \quad -u)^T$$

In a similar way for v

$$\overrightarrow{a_v} \triangleq (0 \quad 0 \quad 0 \quad \tilde{x} \quad \tilde{y} \quad 1 \quad -v\tilde{x} \quad -v\tilde{y} \quad -v)^T$$

Since we have 8 un-known variables, we can define a matrix A for 4 different known points, so h will be the eigen vector if the homogenic linear equation

$$\overline{A} \cdot \vec{h} = 0, \quad where\ \overline{A} = \begin{pmatrix} \overrightarrow{a_{u,1}} \\ \overrightarrow{a_{v,1}} \\ \vdots \\ \overrightarrow{a_{u,4}} \\ \overrightarrow{a_{v,4}} \end{pmatrix}$$

Now, we can see that by solving this homogeneous equations' system, we will get the projective matrix $\overline{H}$.

Since the system is not "noise-free" (and in case more pairs are used), H can be the solution of the optimization problem

$$\arg\min_{\vec{h}} \left|\overline{A} \cdot \vec{h}\right|^2 s.t. |\vec{h}| = 1$$

Note that

$$\left|\overline{A} \cdot \vec{h}\right|^2 = \left(\overrightarrow{Ah}\right)^T \left(\overrightarrow{Ah}\right) = \vec{h}^T \overline{A}^T \overrightarrow{Ah}$$

And denote by $\{\lambda_i\}_{i=1}^9, \{\overline{e_i}\}_{i=1}^9$ the eigen values and eigen vectors, correspondingly, of $A^T A$ such that $A^T A\overline{e_i} = \lambda_i \overline{e_i}$.

Since $A^T A$ is a PSD matrix we know that

$$\lambda_1 \geq \cdots \geq \lambda_9 \geq 0$$

And due to the fact that $\{\overline{e_i}\}_{i=1}^{9}$ forms an orthonormal basis, $\vec{h}$ can be expressed as

$$\vec{h} = \sum_{i=1}^{9} \alpha_i \overline{e_i}$$

Where the norm is,

$$\left|\overrightarrow{h}\right| = 1 = |\overrightarrow{h}|^2 = \left|\sum_{i=1}^{9} \alpha_i \overline{e_i}\right| = \sum_{i=1}^{9} {\alpha_i}^2$$

Therefore,

$$\left|\overline{A} \cdot \vec{h}\right|^2 = \vec{h}^T \overline{A}^T \overline{A} \vec{h} = \sum_{i=1}^{9} \alpha_i \overline{e_i}^T \overline{A}^T \overline{A} \sum_{i=1}^{9} \alpha_i \overline{e_i}$$

After rearrangement and using the fact that $\overline{A}^T \overline{A} \overline{e_i} = \lambda_i \overline{e_i}$ we get that

$$\left|\overline{A} \cdot \vec{h}\right|^2 = \sum_{i=1}^{9} {\alpha_i}^2 \lambda_i \geq \lambda_1 \sum_{i=1}^{9} {\alpha_i}^2 = \lambda_1 = \overline{e_1}^T A^T \overline{A} \overline{e_1} = \left|\overline{A} \overline{e_1}\right|^2$$

$$\xrightarrow{yields} \vec{h}_{opt} = \overline{e_1}$$

In summary, since we have noises in the system the system is not homogenic so we will find the optimal solution of

$$\overline{A}^T \cdot \overline{A}$$

In order to get the solution, we used the mathematic method SVD (Singular Values Decomposition) to get the eigenvectors matrix while the last vector of this matrix should give us the eigenvector for the smallest eigenvalue.

2. See implementation of **compute_homography_naive** in **ex1_student_solution**

3. Running the function from section 2 on the perfect match points file (matches_perfect.mat) retrieves the following normalized Homography matrix, $\overline{H}_{naive}$.

$$\overline{H}_{naive} = \begin{pmatrix} 1.43457214\text{e}+00 & 2.10443232\text{e}-01 & -1.27718679\text{e}+03 \\ 1.34265155\text{e}-02 & 1.34706123\text{e}+00 & -1.60455874\text{e}+01 \\ 3.79279298\text{e}-04 & 5.56523148\text{e}-05 & 1.00000000\text{e}+00 \end{pmatrix}$$

Snapshot for documentation:

```
Naive Homography 0.0005 sec
[[ 1.43457214e+00  2.10443232e-01 -1.27718679e+03]
 [ 1.34265155e-02  1.34706123e+00 -1.60455874e+01]
 [ 3.79279298e-04  5.56523148e-05  1.00000000e+00]]
```

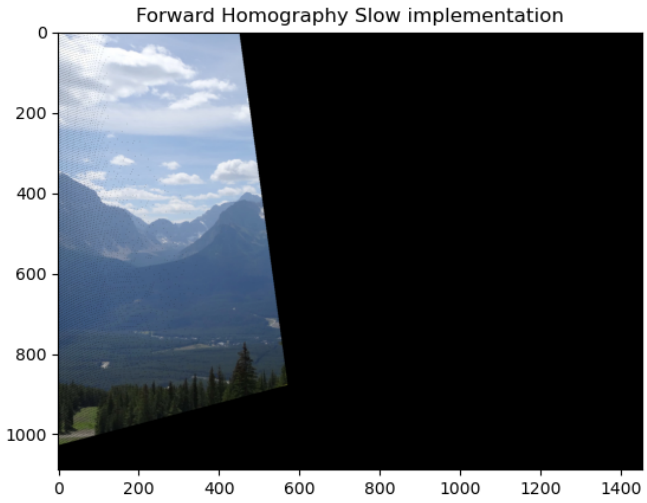
Part A2: Forward Mapping - Slow and Fast

4. See implementation of **compute_forward_homography_slow** in **ex1_student_solution**

The slow commutation took ~9s.

```
Naive Homography Slow computation takes 9.0846 sec
```

The function retrieves the following image.

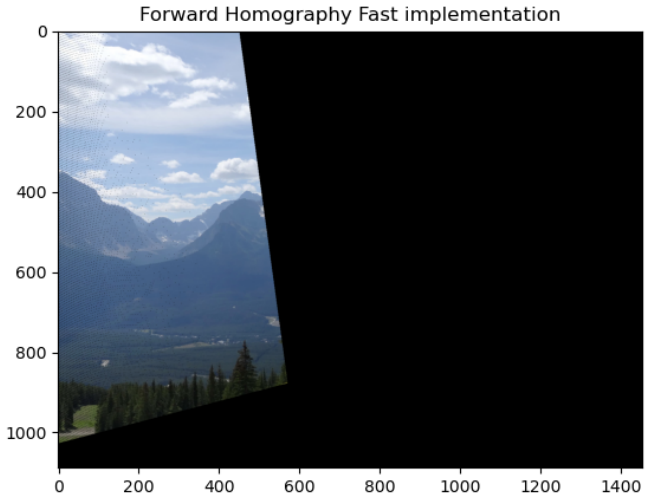


5. See implementation of **compute_forward_homography_fast** in **ex1_student_solution**

The computation time of this method is roughly ~90ms.

```
Naive Homography Fast computation takes 0.0896 sec
```

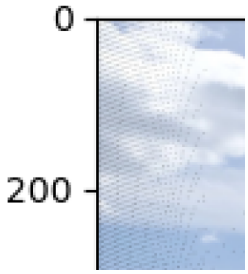
The function retrieves the following image.



It can clearly be observed that the same image was produced by both of the methods of Forward Homography implementation, however the Fast method yields it within much less computation time.

6. Forward Mapping evaluates every pixel in the destination image with the value of its corresponding pixel in the source image, where the corresponding pair of pixels are calculated using the projection matrix H. Therefore, we may encounter the following two problems when using it:
    a. Integer coordinates might be mapped to non-integer coordinates, when the new calculated pixel in the destination image has to be rounded in order to get a valid pixel coordinate.
    b. When the source and destination images are not of the same size or orientation, some information of the source image might be distorted, or missing in the destination image, so that some black pixels might appear in the transformed image.

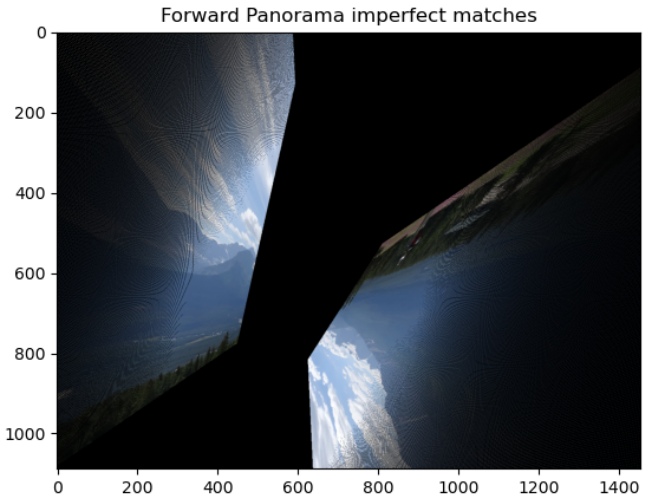
    These issues are seen in the results we yield: dark pixels and artificial edges that are missing in the original image.



7. When we used imperfect match points (matches.mat), we got different Homography matrix.

```
Naive Homography for imperfect matches 0.0004 sec
[[-5.86018363e-01 -1.31259749e-01  6.25479625e+02]
 [-6.25851766e-01 -4.85276766e-01  8.17143196e+02]
 [-9.48023403e-04 -3.85199763e-04  1.00000000e+00]]
```

This projection matrix results in an unclear and messy image that looks even partially upside-down.



As we learned in the class, we know that noises affect the Homography model a lot, so we can conclude that the non-matched points, from imperfect match points file, are the noises that changed our Homography matrix and the result image. In addition, we can see that in the left side of the image the “holes”, that caused by the forward mapping process, were increased.

Part B: Dealing with Outliers

8. See implementation of **test_homography** in **ex1_student_solution**

   The test Homography function calculates the following 2 parameters.
   - Probability that the projection of a single pixel from the source image will be an inlier in the destination image, based on imperfect match points.
   - The mean squared distance between projected pixels from the source image and their actual location in the destination image of inliers.

   Here we can see our results, while the probability is 16% and the MSE error is the other value (456 in pixels units).

```
Naive Homography Fast computation for imperfect matches takes 0.0852 sec
Naive Homography Test 0.0011 sec
[0.16, 456]
```

9. See implementation of **meet_the_model_points** in **ex1_student_solution**

10. For the following RANSAC parameters, let us calculate the given 2 cases.

    $w - inliers\ percent$
    $t - maximal\ error$
    $p - probability\ of\ the\ algorithm\ will\ success$
    $d - the\ minimal\ probability$ of points meeting the model
    $k - number\ of\ iteration$

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

Given: $N = 30,\ w = 80\%$

As we showed in part A, at least 4 points are required in order to find the Homography coefficients. Hence,

$$p = 1 - (1 - 0.8^4)^k$$

We will calculate k in the following cases for by assuming independent random points.

Case I: $p = 90\%$

$$k = \frac{\log(1-0.9)}{\log(1-0.8^4)} = 4.37 \rightarrow\ for\ this\ case\ we\ need\ mimum\ 5\ iterations$$

Case II: $p = 99\%$

$$k = \frac{\log(1-0.99)}{\log(1-0.8^4)} = 8.74 \rightarrow \ for\ this\ case\ we\ need\ mimum\ 9\ iterations$$

For both of the cases introduced in the question, we need at least 9 iterations with 4 random points for each.
In order to cover all the options for selecting $n = 4$ points out of the total 30 matched points, we need $n = \binom{30}{4} = \frac{30!}{4!(30-4)!} = 27405$.

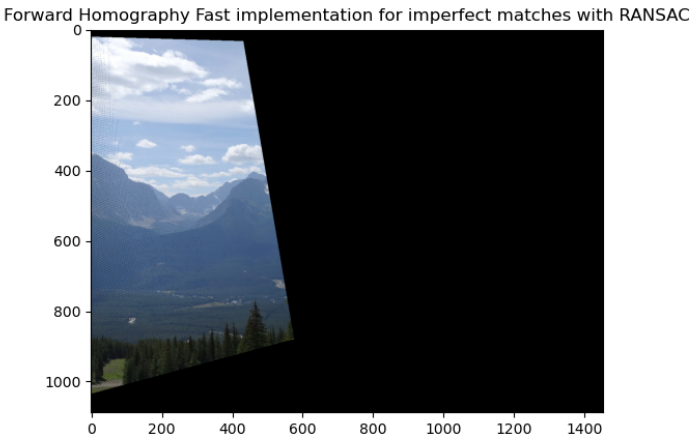
11. See implementation of **compute_homography** in **ex1_student_solution**

12. The obtained RANSAC Homography matrix and its output image are given below.

```
RANSAC Homography 0.0066 sec
[[ 1.43457214e+00  2.10443233e-01 -1.27718679e+03]
 [ 1.34265139e-02  1.34706123e+00 -1.60455867e+01]
 [ 3.79279297e-04  5.56523172e-05  1.00000000e+00]]
RANSAC Homography Test 0.0001 sec
[0.8, 4]
```

Above results show how well the RANSAC algorithm deals with outliers: the projective matrix's coefficients are close to those we received for the perfectly matching points in the previous sections. The MSE distance is also significantly decreased.

All the mentioned above can explains the similarity of the RANSAC output image (below), compared to the perfect match transformation images.



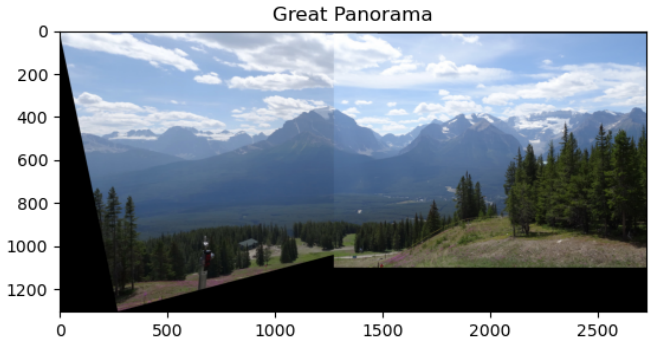
Part C: Panorama Creation

13. See implementation of **compute_backward_mapping** in **ex1_student_solution**

14. See implementation of **add_translation_to_backward_homography** in **ex1_student_solution**

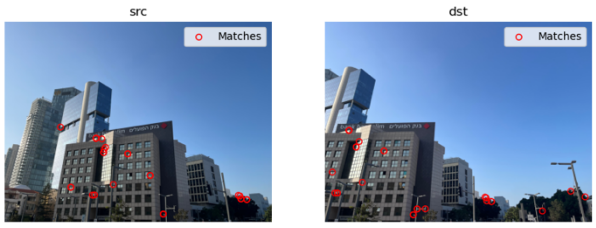
15. See implementation of **panorama** in **ex1_student_solution**

16. Panorama output image



The images are "stitched" thanks to the overlapping sections where the inliers are.

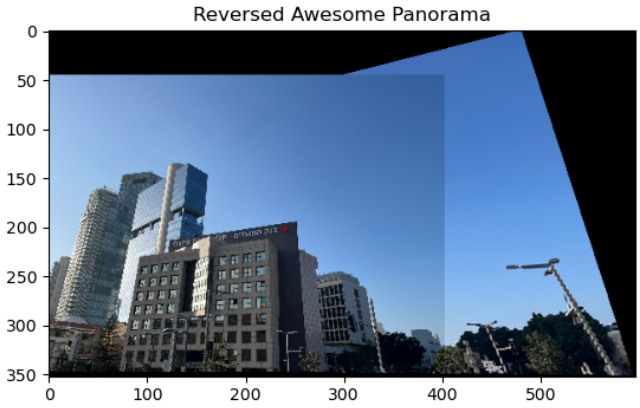
17. In order to get acceptable results, we chose the matching points that marked in red below.

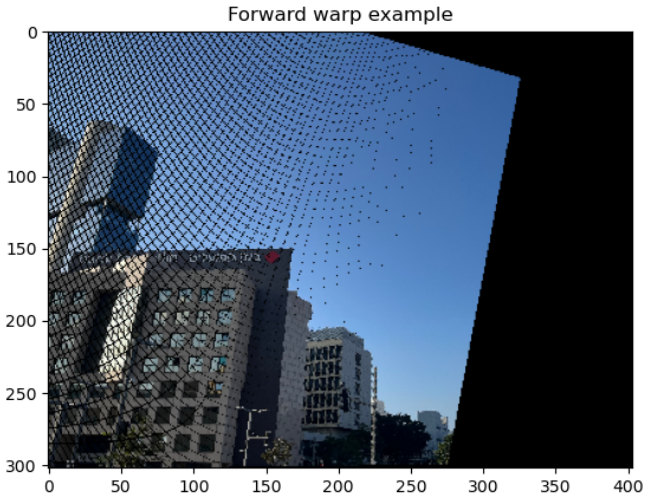


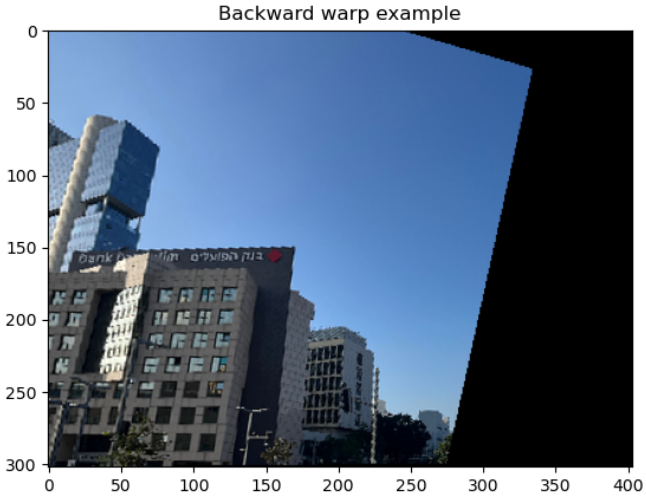
In addition, we changed the maximal error and decimation factor as shown below.

```
DECIMATION_FACTOR = 5.0
max_err = 28  # <<<<< YOU MAY CHANGE THIS
inliers_percent = 0.8  # <<<<< YOU MAY CHANGE THIS
```

And then ran the script and got the following results. While we can see the we got the same results with similar phenomena as we got previously above.




Forward warp example
0
50
100
150
200
250
300
0
50
100
150
200
250
300
350
400


Backward warp example
0
50
100
150
200
250
300
0
50
100
150
200
250
300
350
400