ECE661 Fall 2024: Homework 2

By – Aayush Bhadani

abhadani@purdue.edu

TASK 1

The goal is to first find the homography between a pair of images and then project the region of interest from one image on the demarcated region in the other image. We do this by picking a set of 4 corresponding points on images and using them to find the elements of the homography matrix (H) that relate the images to each other.

We know that two points X and X' on domain and range plane respectively are related by the Homography matrix(H) with the equation as:

$$X = HX'$$

(Homography matrix is 3x3 non-singular & the points are in HC form rep. as 3D vectors)

We can also write it as:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & v \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

In physical representation of point in domain plane, $x = x_1/x_3$ and $y = x_2/x_3$. We can set $x_3 = 1$ to get a direct relation of $x = x_1$ and $y = x_2$.

Similarly for the range plane point, $x' = x_1'/x_3'$ and $y' = x_2'/x_3'$.

We can also set $v = 1$.

So we get,

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

Expanding:

$$x_1' = a_{11}x_1 + a_{12}x_2 + t_x$$

$$x_2' = a_{21}x_1 + a_{22}x_2 + t_y$$

$$x_3' = v_1x_1 + v_2x_2 + 1$$

We know

$$x' = \frac{x_1'}{x_3'} = \frac{a_{11}x_1 + a_{12}x_2 + t_x}{v_1x_1 + v_2x_2 + 1} = \frac{a_{11}x + a_{12}y + t_x}{v_1x + v_2y + 1}$$

and

$$y' = \frac{x_2'}{x_3'} = \frac{a_{21}x_1 + a_{22}x_2 + t_y}{v_1x_1 + v_2x_2 + 1} = \frac{a_{21}x + a_{22}y + t_y}{v_1x + v_2y + 1}$$

Simplifying this we get,

$$x' = a_{11}x + a_{12}y + t_x - v_1xx' - v_2yx'$$

$$y' = a_{21}x + a_{22}y + t_y - v_1xy' - v_2yy'$$

We get 2 equations corresponding to a pair of points. Similarly, for 4 such pairs of points(denoted with subscript i=1,2,3,4), we will have 8 equations. Using this we can find the 8 unknowns in the Homography matrix H. Matrix representation of these equations is given below:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & x_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{pmatrix}$$

This is of the form $Ax = b$. Vector containing unknown elements of H is given by $x = A^{-1}b$. Finally, we plug the solved values of unknowns in H to get our required homography matrix.

For Affine homography we have $H = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$

We get the matrix formulation as,

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \\ x_4 & y_4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_4 & x_4 & 1 \end{bmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{pmatrix}$$

Again, this is of the form $Ax = b$ from which we get $x = A^{-1}b$. Here we would have to do pseudo inverse as the matrix is not square.

(In another approach we could have used just 3 corresponding pair points which would give 6 equations, sufficient to find 6 unknowns of affine homography matrix by taking inverse of 6x6 matrix)

We start the task by selecting 4 points on the domain image that demarcates the region of interest. Similarly, we also select 4 points P, Q, R, S on the range image (outer border of frame) on which we will project.

For picking points, python based tool was used which is described in "Displaying the coordinates of the points clicked on the image using Python-OpenCV." https://www.geeksforgeeks.org/displaying-the-coordinates-of-the-points-clicked-on-the-image-using-python-opencv/. Although, the GIMP tool was also explored.

Next, we find the homography matrix for a given pair of domain and the range images as per the equations shown above. After we have formed the homography matrix (H) by finding the unknown elements we perform the projection.

To get the projection, the inverse homography matrix $H^{-1}$is used which maps $X = H^{-1}X'$.

We go over the coordinates of the range image and get the corresponding transformed coordinates (typecasted to int) of the domain plane using H^{-1} transform. We next check if this obtained domain plane coordinate lies in the region of interest of our domain image. If it does, we update the pixel value of the coordinate in the range image with the corresponding pixel value from the domain image.

1.1

4 points are selected on each of the provided images starting from upper left corner and going anticlockwise (P,S,R,Q order).

Image 1:

Points:

x1, y1 = 422,856

x2, y2 = 680,3158

x3, y3 = 2390,2280

x4, y4 = 2541,877

Image 2:

Points:

x1, y1 = 521,1437

x2, y2 = 465,2674

x3, y3 = 1934,2801

x4, y4 = 1856,824

Image 3:

Points:

x1, y1 = 1206,548

x2, y2 = 249,1785

x3, y3 = 1783,3167

x4, y4 = 2955,2293

Image 4:

Points:

x1, y1 = 1,162

x2, y2 = 1,655

x3, y3 = 780,655

x4, y4 = 780,162

We take region of interest from Image 4 and project it on the PQRS frame (demarcated by 4 points) in each on the other 3 images. For this we find the homographies between Image 4 and other 3 images (Image 1, Image 2, Image 3) one at a time.

The projection results are as follows:

Image4 -> Image1

Image4 -> Image2

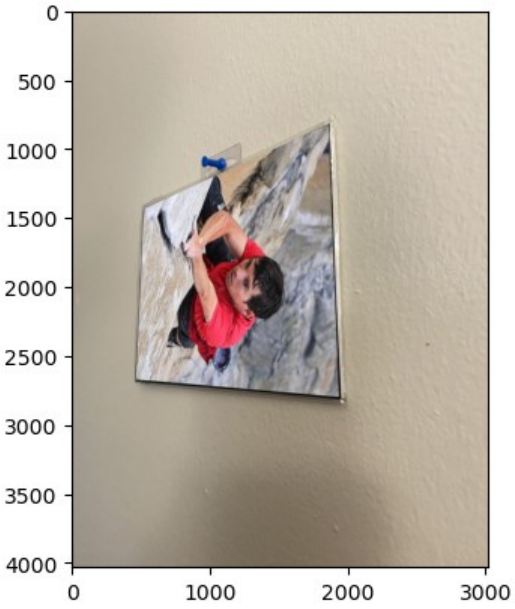
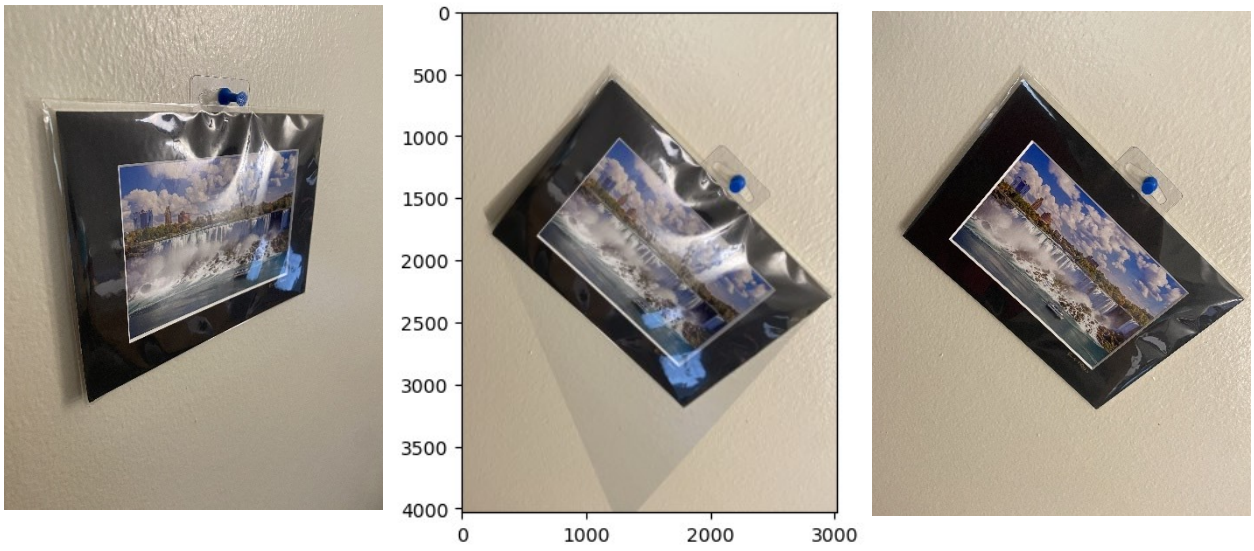


Image4 -> Image3

1.2

We first get homography between Image 1 & Image 2. Next, we get homography between Image2 & Image3. Applying the product of these two homographies to Image1 we get a result which is similar to Image3 which is shown below:

Image1 result image Image3



1.3

Now we use **affine homography** (discussed above) and map from Image4 to PQRS frame in Image1, Image2 & Image3 respectively. The results are shown below:

Image4 -> Image1 Image4 -> Image2

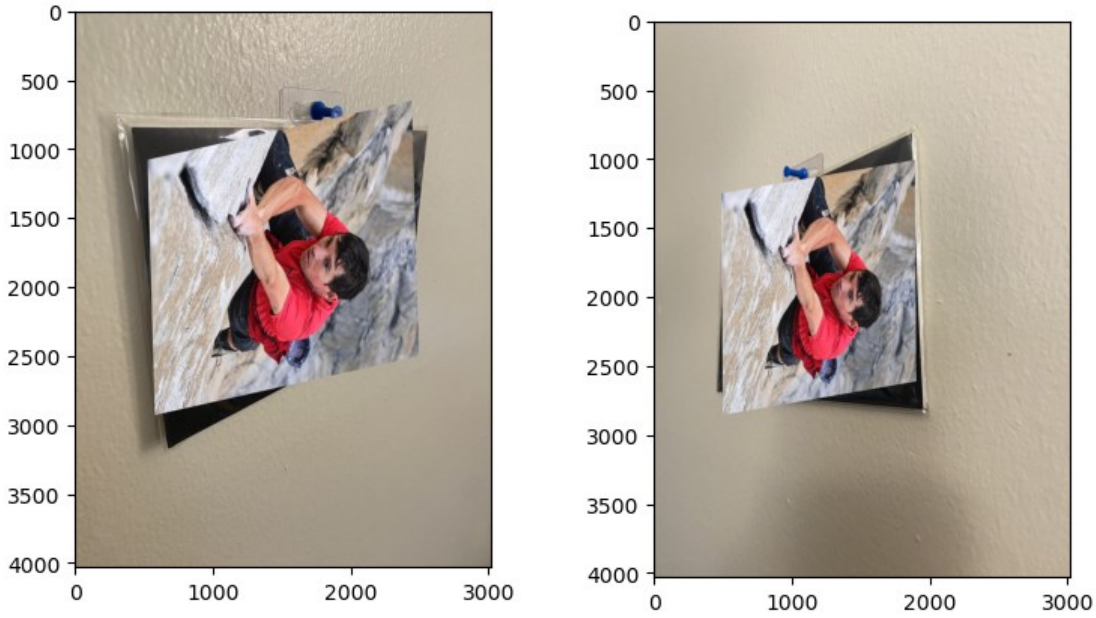


Image4 -> Image3

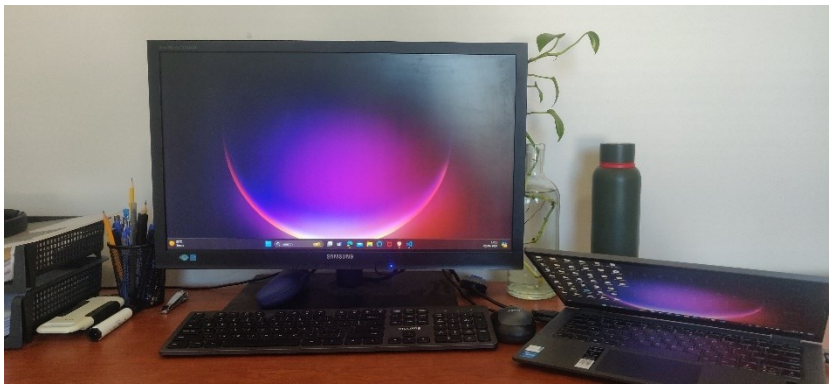
The third result where we map from Image4 to image3 is better than others. This can be attributed to the fact that affine homography keeps parallel lines parallel. The points selected in first two cases do not produce perfectly parallel lines and thus produce inferior results compared to the third case where lines are almost parallel.

TASK 2

Task1 is repeated with own images that are shown below along with the selected points.

Corners (inner) of the monitor screen are selected as points starting from upper left corner and going anti-clockwise (task bar at bottom excluded). Goal is to change the monitor wallpaper using projection.

Image1



Points:

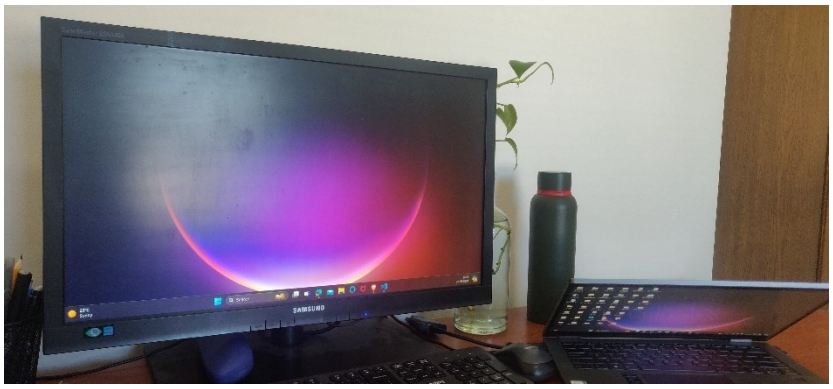
x1, y1 = 873,263

x2, y2 = 912,1292

x3, y3 = 2836,1297

x4, y4 = 2855,278

Image2



Points:

x1, y1 = 327,175

x2, y2 = 331,1677

x3, y3 = 2665,1482

x4, y4 = 2694,414

Image3

Points:

x1, y1 = 2055,760

x2, y2 = 2089,1482

x3, y3 = 3446,1575

x4, y4 = 3412,629

Image4

Points:

x1, y1 = 8,8

x2, y2 = 8,3679

x3, y3 = 5562,3679

x4, y4 = 5553,17

2.1

Projections:

Image4 -> Image1

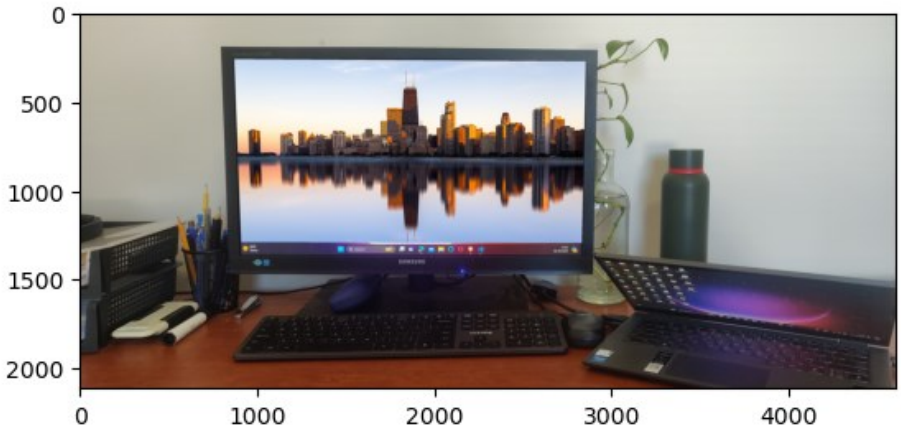


Image4 -> Image2

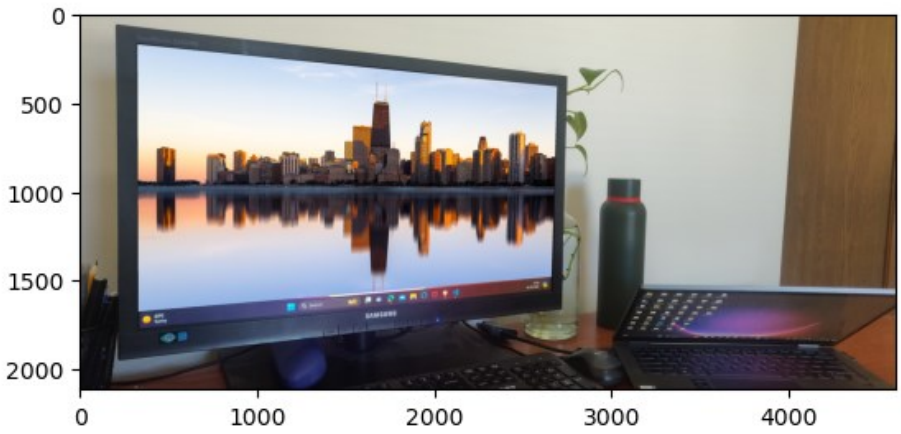
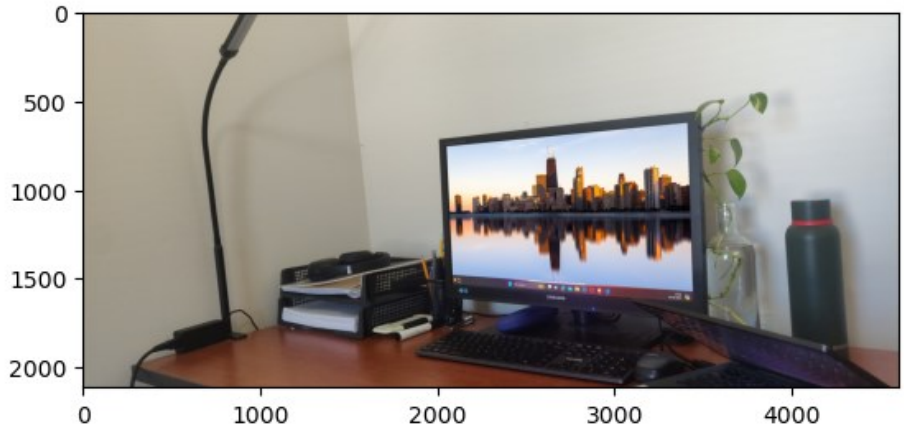
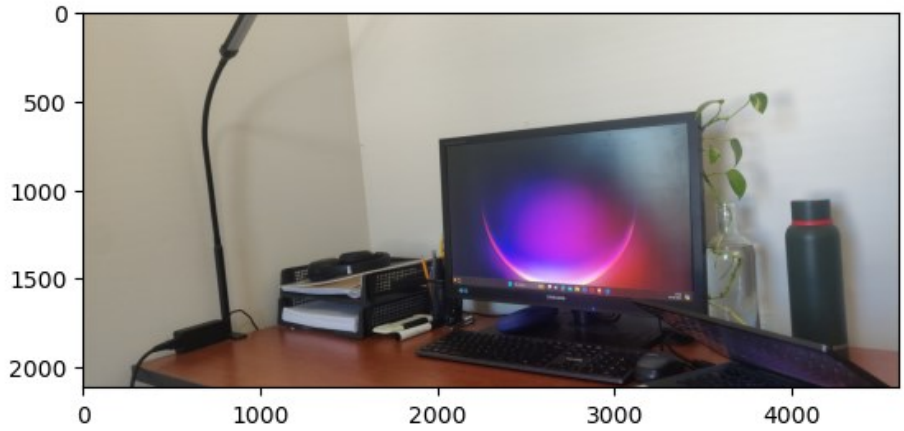


Image4 -> Image3



2.2

Result image after applying the product of these two homographies to Image1 which is similar to Image3



2.3 Using affine homographies.

Image4 -> Image1

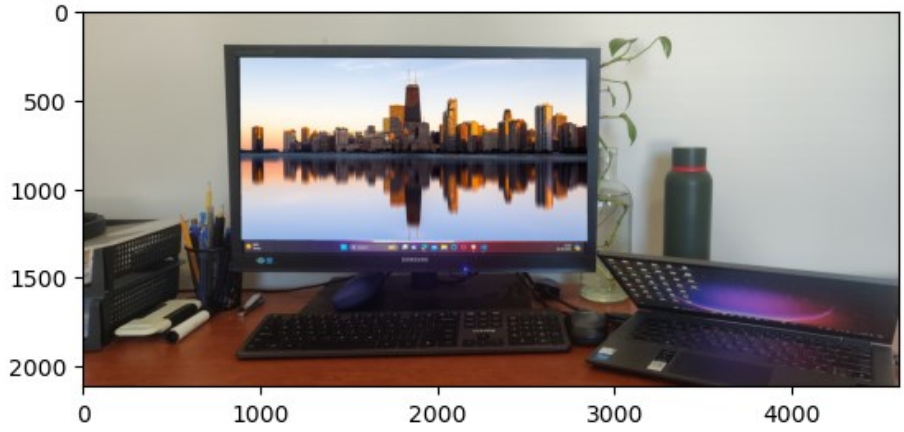


Image4 -> Image2

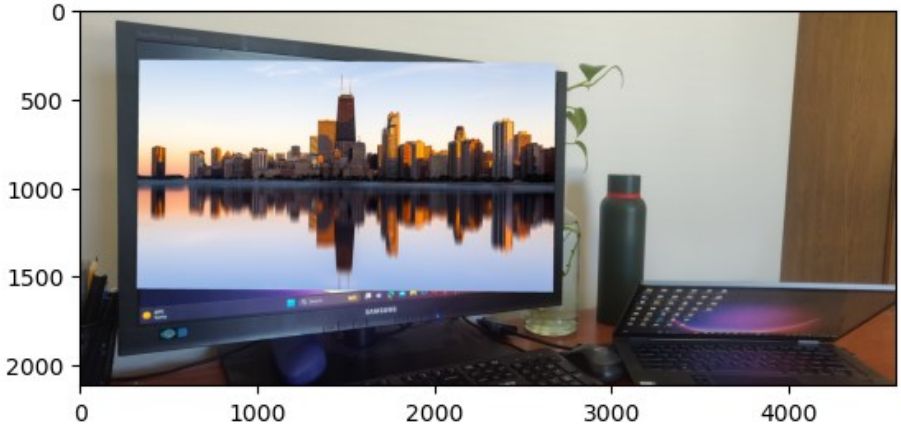
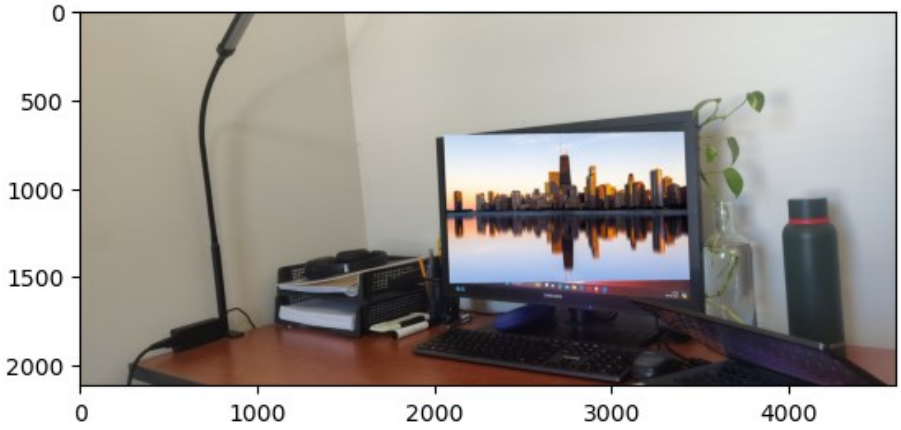


Image4 -> Image3



Here the first result is better than the other two as lines are parallel in that case.

Extra:

1.

Homography for Rotating:

$$H(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Homography for vertical tilting:

$$H(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix}$$

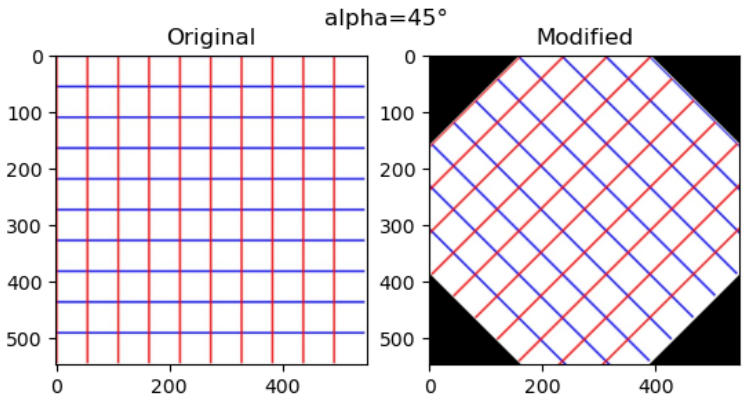
Homography for horizontal tilting:

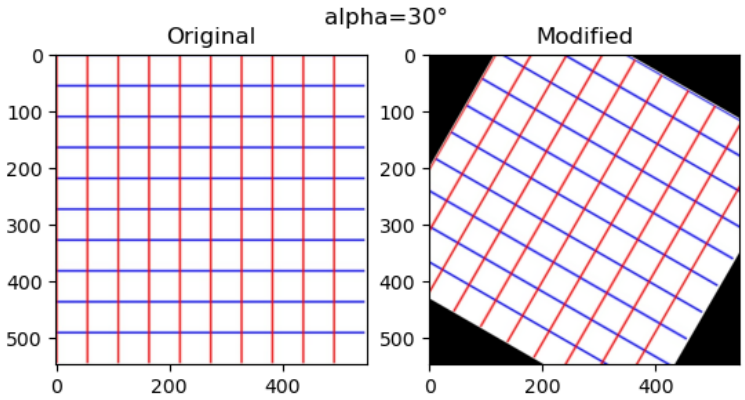
$$H(\alpha) = \begin{bmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{bmatrix}$$

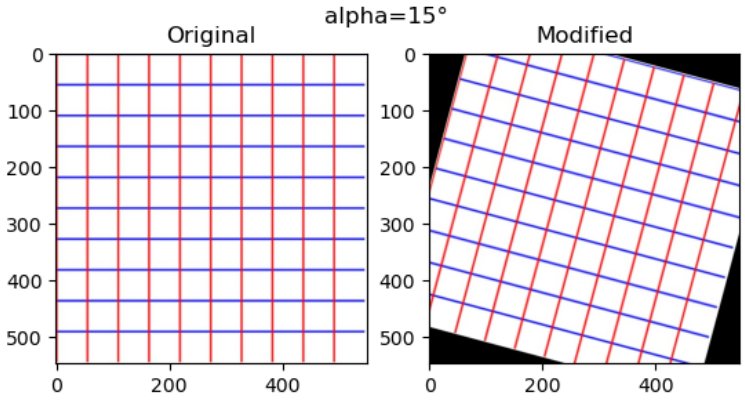
2.

Results after applying homographies are shown below.

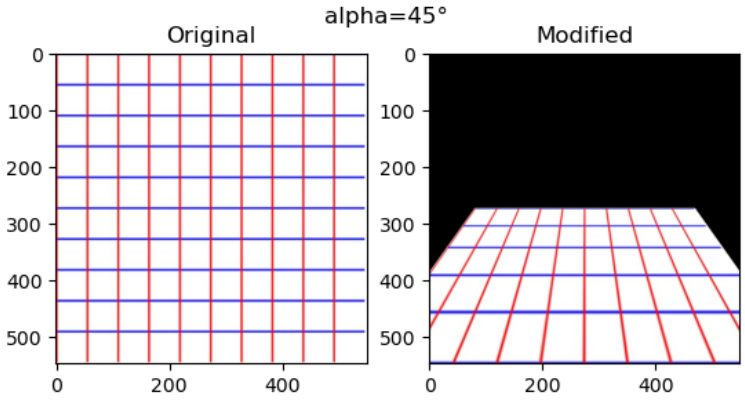
For rotation:

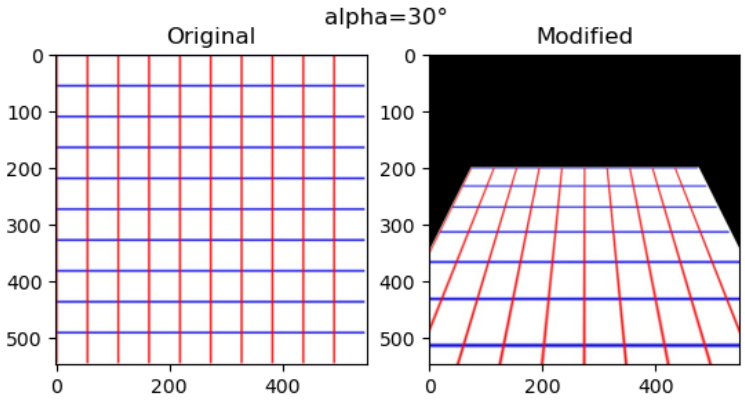


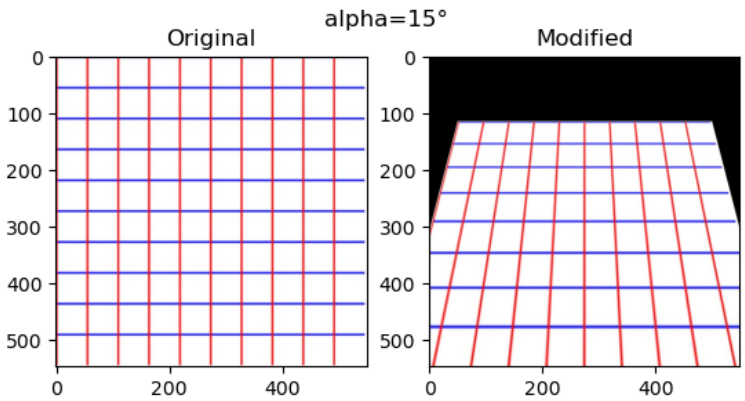




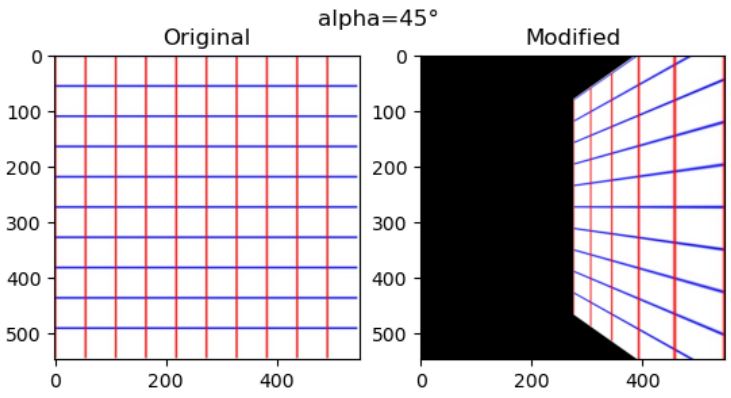
For vertical tilting:

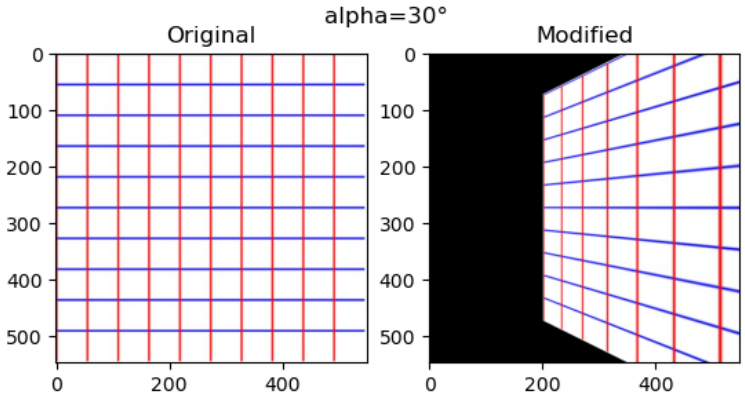


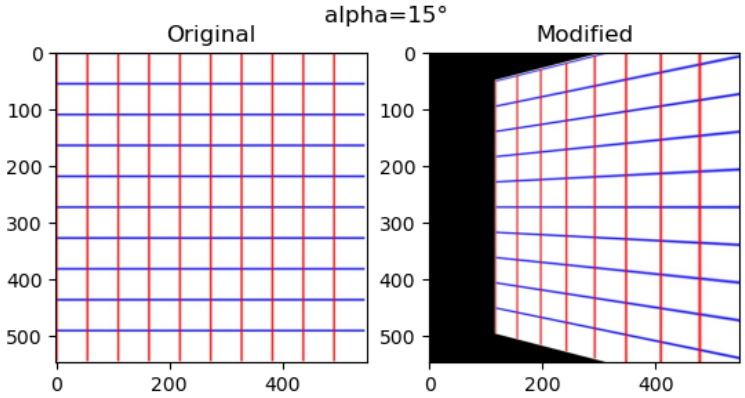




For horizontal tilting:







3.

Under vertical tilting horizontal line remains invariant, whereas under horizontal tilting it is the vertical line which is invariant.

If we consider vertical tilting,

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} = H^{-T}$$

A horizontal line can be represented as $l = \begin{pmatrix} 0 \\ x_2 \\ x_3 \end{pmatrix}$

And a vertical line can be represented as $l = \begin{pmatrix} x_1 \\ 0 \\ x_3 \end{pmatrix}$

Performing transform $l' = H^{-T}l$

Horizontal line transform: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} \begin{pmatrix} 0 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ x_2 cos\alpha + x_3 sin\alpha \\ -x_2 sin\alpha + x_3 cos\alpha \end{pmatrix}$

We get a horizontal line as result

Vertical line transform: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} \begin{pmatrix} x_1 \\ 0 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_3 sin\alpha \\ x_3 cos\alpha \end{pmatrix}$

Result is not a vertical line.

Similarly, we perform analysis for <u>horizontal tilting</u>:

$$H = \begin{bmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{bmatrix} = H^{-T}$$

Performing transform $l' = H^{-T} l$ with vertical and horizontal lines

Horizontal line transform: $\begin{bmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{bmatrix} \begin{pmatrix} 0 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_3 sin\alpha \\ x_2 \\ x_3 cos\alpha \end{pmatrix}$

Result is not a horizontal line.

Vertical line transform: $\begin{bmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{bmatrix} \begin{pmatrix} x_1 \\ 0 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 cos\alpha + x_3 sin\alpha \\ 0 \\ -x_1 sin\alpha + x_3 cos\alpha \end{pmatrix}$

We can also show this numerically,

horizontal_line = [[0., 0., 0., 0., 0., 0., 0., 0., 0.],

[1., 1., 1., 1., 1., 1., 1., 1., 1.],

[-5.,-4.,-3.,-2.,-1., 0., 1., 2., 3., 4.]]

vertical_line = [[1., 1., 1., 1., 1., 1., 1., 1., 1.],

[0., 0., 0., 0., 0., 0., 0., 0., 0.],

[-5.,-4.,-3.,-2.,-1., 0., 1., 2., 3., 4.]]

Performing vertical tilting (angle 30 deg) on

horizontal line we get:

```
[[  0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.       ]
 [  1.35688464  1.35871279  1.36054094  1.3623691   1.36419725
    1.3660254   1.36785356  1.36968171  1.37150986  1.37333802]
 [-275.33012702 -274.96410162 -274.59807621 -274.23205081 -273.8660254
  -273.5       -273.1339746  -272.76794919 -272.40192379 -272.03589838]]
```

vertical line we get:

```
[[  1.          1.          1.          1.          1.
    1.          1.          1.          1.          1.       ]
 [  0.49268739  0.49451554  0.49634369  0.49817185  0.5
    0.50182815  0.50365631  0.50548446  0.50731261  0.50914077]
 [-175.85615368 -175.49012828 -175.12410287 -174.75807747 -174.39205206
  -174.02602666 -173.66000126 -173.29397585 -172.92795045 -172.56192505]]
```

Performing horizontal tilting (angle 30 deg) on

horizontal line we get:

```
[[  0.48907104  0.49089253  0.49271403  0.49453552  0.49635701
0.49817851  0.5         0.50182149  0.50364299  0.50546448]
[  1.          1.          1.          1.          1.
1.          1.          1.          1.          1.       ]
[-175.22217908 -174.85615368 -174.49012828 -174.12410287 -173.75807747
-173.39205206 -173.02602666 -172.66000126 -172.29397585 -171.92795045]]
```

vertical line we get:

```
[[ 1.35691794e+00  1.35873943e+00  1.36056092e+00  1.36238242e+00
1.36420391e+00  1.36602540e+00  1.36784690e+00  1.36966839e+00
1.37148988e+00  1.37331138e+00]
 [-1.09613564e-19 -8.76908514e-20 -6.57681385e-20 -4.38454257e-20
-2.19227128e-20  0.00000000e+00  2.19227128e-20  4.38454257e-20
6.57681385e-20  8.76908514e-20]
[-2.76330127e+02 -2.75964102e+02 -2.75598076e+02 -2.75232051e+02
-2.74866025e+02 -2.74500000e+02 -2.74133975e+02 -2.73767949e+02
-2.73401924e+02 -2.73035898e+02]]
```