Prateek Gupta
gupta198@purdue.edu
ECE 661 Homework 2

Overview

A homography is a mapping from one plane to another. We do this by multiplying the homography (a 3x3 matrix) by each coordinate in the representational space on the plane we are interested in transforming. We mut first find the homography by selecting a few points on a source image and the corresponding points they will be mapped to on the destination image. Once the homography is calculated, it can be applied to each point on the source image. This will result tell us where to put that pixel on the destination image. We do this to every pixel on the source. The result is a distorted image that fits in the intended way on the destination.

Calculating the Homography

The homography is calculated by selecting at least four mappings of points. So, we pick a point on the image we want to distort, and a corresponding point on the image we want to map to. We need to do this at least four times so that we can create enough equations for the number of unknowns we will have. The easiest points to select are the corners. We will find the coordinates of the corners, and where we want those points to end up.

$(x_1, y_1) \rightarrow (x_1', y_1')$
$(x_2, y_2) \rightarrow (x_2', y_2')$
$(x_3, y_3) \rightarrow (x_3', y_3')$
$(x_4, y_4) \rightarrow (x_4', y_4')$

Using these points, we will calculate the homography H

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

The homography H can be multiplied to all points in the 2-dimensional representational space to get the distorted image. The vector *x* will represent the original point, and the vector *x'* will represent the distorted point. We get *x'* by multiplying *x* with the homography H.

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

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad x' = \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix}$$

Using the equation for finding the distorted points, we can construct a system of three equations:

$$x_1' = h_{11}x_1 + h_{12}x_2 + h_{13}x_3$$
$$x_2' = h_{21}x_1 + h_{22}x_2 + h_{23}x_3$$
$$x_3' = h_{31}x_1 + h_{32}x_2 + h_{33}x_3$$

Now, in order to find the coordinates of the distorted point, we are only interested in the ratios. So, the distorted x coordinate (x') can be found by computing the ratio between x_1 and x_3, and the distorted y coordinate (y') can be found by computing the ratio between x_2 and x_3.

$$x' = \frac{x_1'}{x_3'} = \frac{h_{11}x_1 + h_{12}x_2 + h_{13}x_3}{h_{31}x_1 + h_{32}x_2 + h_{33}x_3}$$

$$y' = \frac{x_1'}{x_3'} = \frac{h_{21}x_1 + h_{22}x_2 + h_{23}x_3}{h_{31}x_1 + h_{32}x_2 + h_{33}x_3}$$

Again, since we are only concerned with the ratios, we can set x_3 and h_{33} to 1. We will also change x_1 to x and x_2 to y, so that we can accommodate multiple sets of equations without getting too confused with the subscripts

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

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

And we can manipulate these algebraically to get

$$x' = h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx'$$
$$y' = h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy'$$

In order to solve for all values h, we need to utilize four pairs of points to construct 4 sets of equations, giving us a total of 8 equations with 8 unknowns. This can be represented by

$$\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}$$

We can now solve for all values h easily by plugging in the four pairs of points and using matrix algebra.

$$\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix}^{-1} \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}$$

This will give us the homography matrix

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

Applying the Homography

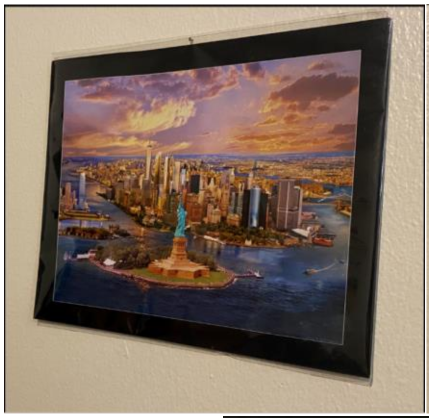
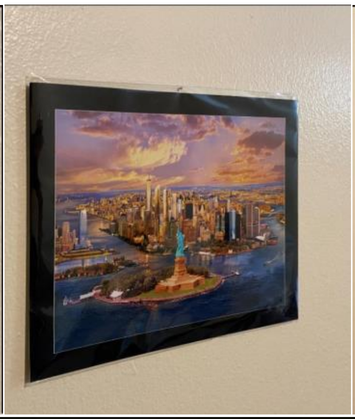
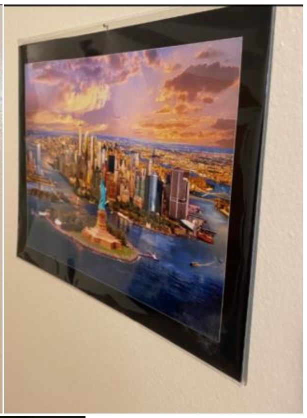
We then apply this homography to every point in the representational space on the image we want to distort. Since we are concerned with the ratios, we will augment 1 onto each point. We will multiply the matrices and get the value of the distorted point in the representational space

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

To convert this into a point in the physical space, we will divide x_1 and x_2 by x_3. This will give us the x and y coordinates to map the point to. We will then replace the pixel in the destination image with the value we just calculated.

Task 1.1

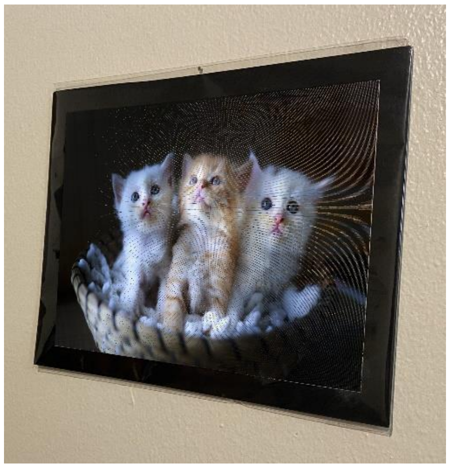
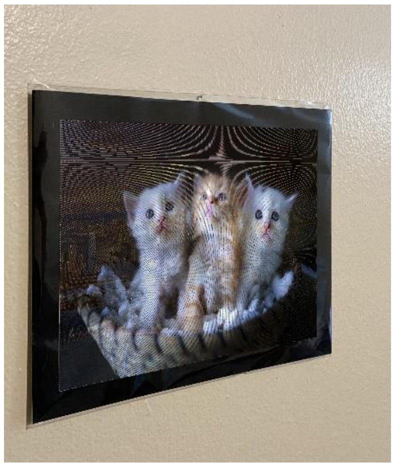
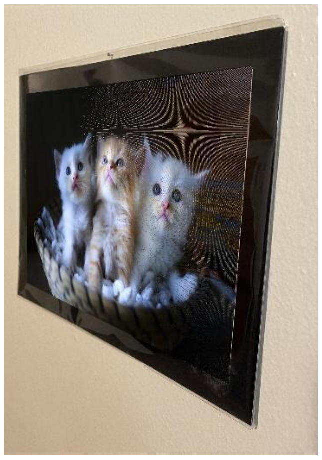
We will apply the methodology described above to superimpose the last image onto the painting shown in the first 3

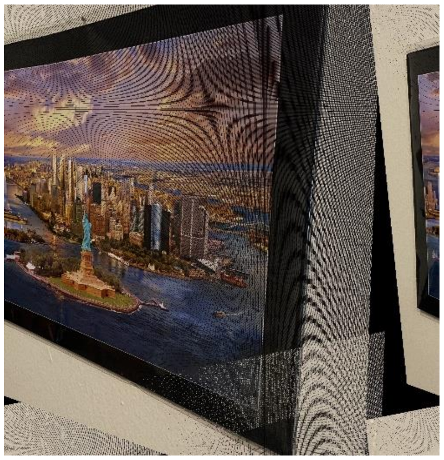
As described in the methodology we want to use the coordinates of the corners of the last image and the corresponding coordinates on each painting. The coordinates can be found by loading each image into GIMP and hovering over them with the cursor. We will use the following mapping:

Painting 1:	*Painting 2:*	*Painting 3:*
(0, 0) → (209, 511)	(0, 0) → (341, 695)	(0, 0) → (108, 440)
(0, 1125) → (239, 1607)	(0, 1125) → (333, 2330)	(0, 1125) → (117, 1369)
(1920, 1125) → (1686, 1828)	(1920, 1125) → (1884, 2003)	(1920, 1125) → (1100, 1864)
(1920, 0) → (1775, 357)	(1920, 0) → (1887, 753)	(1920, 0) → (1218, 303)

Doing this, we get the following images:

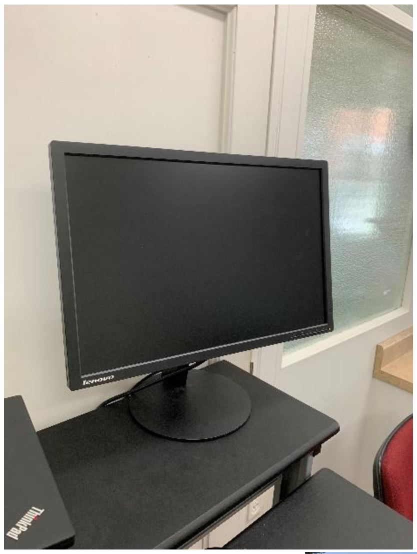
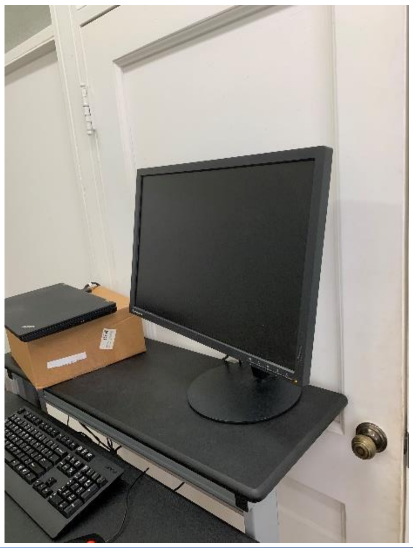
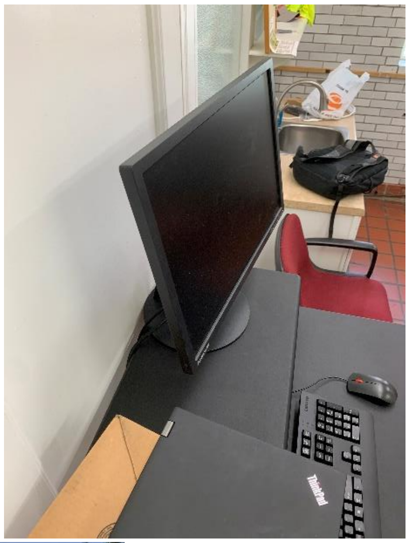




Task 1.2
In this task, we find the homography of image 1 on image 2, and of image 2 on image 3, then multiply them. We then apply this new homography to image 1. This is the result:



Task 2

In this task we will use some original pictures. They are shown below.

We will map the fourth image onto the monitor in the first three. We will use the following points to calculate the homography

Monitor 1:	*Monitor 2:*	*Monitor 3:*
(0, 0) → (454, 1117)	(0, 0) → (1029, 1289)	(0, 0) → (1045, 1281)
(0, 3024) → (576, 2744)	(0, 3024) → (975, 2259)	(0, 3024) → (1439, 2663)
(4032, 3024) → (2384, 2357)	(4032, 3024) → (2166, 2742)	(4032, 3024) → (2079, 1530)
(4032, 0) → (2331, 1229)	(4032, 0) → (2321, 1165)	(4032, 0) → (1984, 499)

The output is shown below (also note that light on the monitor is orange and in standby mode. It is not actually displaying the image):

lenovo
ThinkPad

ThinkPad

Source Code:

```
import cv2
import numpy as np

#get the homography between two images
def getHomography(im1Pts, im2Pts):
    px = im1Pts[0, 1]
    py = im1Pts[0, 0]
    qx = im1Pts[1, 1]
    qy = im1Pts[1, 0]
    rx = im1Pts[2, 1]
    ry = im1Pts[2, 0]
    sx = im1Pts[3, 1]
    sy = im1Pts[3, 0]

    pxp = im2Pts[0, 1]
    pyp = im2Pts[0, 0]
    qxp = im2Pts[1, 1]
    qyp = im2Pts[1, 0]
    rxp = im2Pts[2, 1]
    ryp = im2Pts[2, 0]
    sxp = im2Pts[3, 1]
    syp = im2Pts[3, 0]

    t = np.array([[pxp],[pyp],[qxp],[qyp],[rxp],[ryp],[sxp],[syp]])
    P = np.array([[px, py, 1, 0, 0, 0, -px*pxp, -py*pxp],
                 [0, 0, 0, px, py, 1, -px*pyp, -py*pyp],
                 [qx, qy, 1, 0, 0, 0, -qx*qxp, -qy*qxp],
                 [0, 0, 0, qx, qy, 1, -qx*qyp, -qy*qyp],
                 [rx, ry, 1, 0, 0, 0, -rx*rxp, -ry*rxp],
                 [0, 0, 0, rx, ry, 1, -rx*ryp, -ry*ryp],
                 [sx, sy, 1, 0, 0, 0, -sx*sxp, -sy*sxp],
                 [0, 0, 0, sx, sy, 1, -sx*syp, -sy*syp]])
    pInv = np.linalg.inv(P)
    h = pInv.dot(t)
    H = np.array([[h[0, 0], h[1, 0], h[2, 0]], [h[3, 0], h[4, 0], h[5, 0]], [h[6, 0], h[7, 0], 1]])
    return H

#get the resulting image by applying the homgraphy
def newImage(srcIm, destIm, corners, H):
    for row in range(srcIm.shape[0]):
        for col in range(srcIm.shape[1]):
            coord = np.matmul(H, [[row], [col], [1]])
            coord /= coord[2, 0]
            if coord[0, 0] < srcIm.shape[0] and coord[1, 0] < srcIm.shape[1]:
                destIm[int(coord[0][0])][int(coord[1][0])] = srcIm[row][col]
    return destIm

def task1_1():
    catCorners = np.array([[0, 0], [0, 1125], [1920, 1125], [1920, 0]])
    im1Corners = np.array([[299, 511], [239, 1607], [1686, 1828], [1775, 357]])
    im2Corners = np.array([[341, 695], [333, 2330], [1884, 2003], [1887, 753]])
    im3Corners = np.array([[108, 440], [117, 1369], [1100, 1864], [1218, 303]])

    H1 = getHomography(catCorners, im1Corners)
    H2 = getHomography(catCorners, im2Corners)
    H3 = getHomography(catCorners, im3Corners)

    im1 = cv2.imread('Task1_Images/painting1.jpeg')
    im2 = cv2.imread('Task1_Images/painting2.jpeg')
    im3 = cv2.imread('Task1_Images/painting3.jpeg')
    imCat = cv2.imread('Task1_Images/kittens.jpeg')

    im1New = newImage(imCat, im1, im1Corners, H1)
    cv2.imwrite('Task1_Results/Image1.jpeg', im1New)
    im2New = newImage(imCat, im2, im2Corners, H2)
    cv2.imwrite('Task1_Results/Image2.jpeg', im2New)
    im3New = newImage(imCat, im3, im3Corners, H3)
    cv2.imwrite('Task1_Results/Image3.jpeg', im3New)
    return

def task1_2():
    im1Corners = np.array([[299, 511], [239, 1607], [1686, 1828], [1775, 357]])
    im2Corners = np.array([[341, 695], [333, 2330], [1884, 2003], [1887, 753]])
    im3Corners = np.array([[108, 440], [117, 1369], [1100, 1864], [1218, 303]])

    H1 = getHomography(im1Corners, im2Corners)
    H2 = getHomography(im2Corners, im3Corners)
    H = np.matmul(H1, H2)
    H /= H[2, 2]

    img = cv2.imread('Task1_Images/painting1.jpeg')
    black = np.zeros((img.shape[0], img.shape[1], 3),dtype='uint8')
    corners = np.array([[0, 0], [0, 1125], [1920, 1125], [1920, 0]])
    cv2.imwrite('black.jpg', black)
    imNew = newImage(img, black, im3Corners, H)
    cv2.imwrite('Task1_Results/Task1_2.jpeg', imNew)
    return

def task2():
    mountCorners = np.array([[0, 0], [0, 3024], [4032, 3024], [4032, 0]])
    im1Corners = np.array([[454, 1117], [576, 2744], [2384, 2357], [2331, 1229]])
    im2Corners = np.array([[1029, 1289], [975, 2259], [2166, 2742], [2321, 1165]])
    im3Corners = np.array([[1045, 1281], [1439, 2663], [2079, 1530], [1984, 499]])

    H1 = getHomography(mountCorners, im1Corners)
    H2 = getHomography(mountCorners, im2Corners)
    H3 = getHomography(mountCorners, im3Corners)

    im1 = cv2.imread('Task2_Images/1.jpg')
    im2 = cv2.imread('Task2_Images/2.jpg')
    im3 = cv2.imread('Task2_Images/3.jpg')
    imMt = cv2.imread('Task2_Images/4.jpg')

    im1New = newImage(imMt, im1, im1Corners, H1)
    cv2.imwrite('Task2_Results/Image1.jpg', im1New)
    im2New = newImage(imMt, im2, im2Corners, H2)
    cv2.imwrite('Task2_Results/Image2.jpg', im2New)
    im3New = newImage(imMt, im3, im3Corners, H3)
    cv2.imwrite('Task2_Results/Image3.jpg', im3New)
    return

task1_1()
task1_2()
task2()
```