Assignment 2

ECE 181 - Introduction to Computer vision

Kenny Hoang Nguyen
X299187
Section #3

January 2020

1 Discussion

First of all we had to mark points and get the coordinates in each picture taken from the two different angles. The points had to be the same points at both pictures. The points are in the picture, but since the font was small I chose to list the points in the tables. The points are counted from upper left corner in the windows, then going counter-clockwise. point 1-4 is the leftmost window and point 5-8 is the rightmost window. I chose these windows because they have distinct edges that are present in both pictures, and I wanted some distance between the points as well so I chose the two windows furthest away from each other.

1.a Picture of school

Doing the transformation on the picture taken from the right resulted in the picture in figure 3 These points resulted in the transformation matrix in table 3 Note that the transformed picture still has alot of the same details, but the glare in the original picture taken from right is not there.

Points	X	Y
Point 1	148.29	320.65
Point 2	149.02	421.83
Point 3	213.72	428.68
Point 4	212.96	332.82
Point 5	1399.79	506.28
Point 6	1402.84	577.79
Point 7	1437.07	581.60
Point 8	1430.99	510.84

Table 1: For picture of school taken from the left

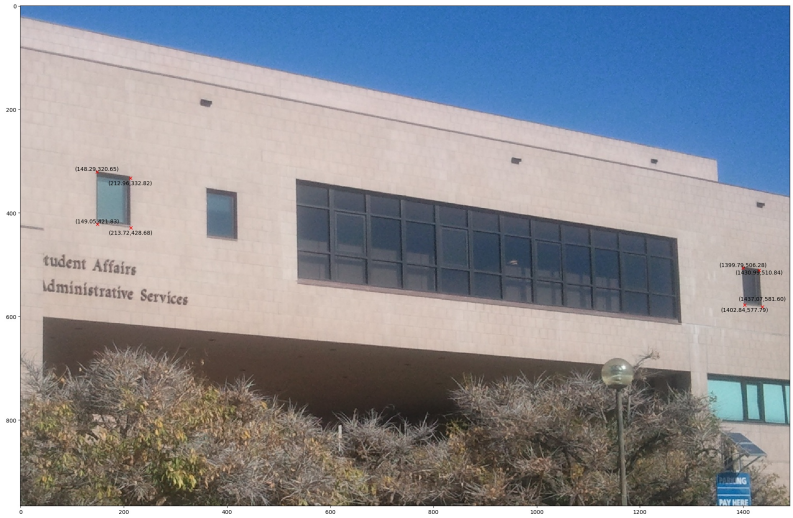

Figure 1: The coordinates are listed in table 1

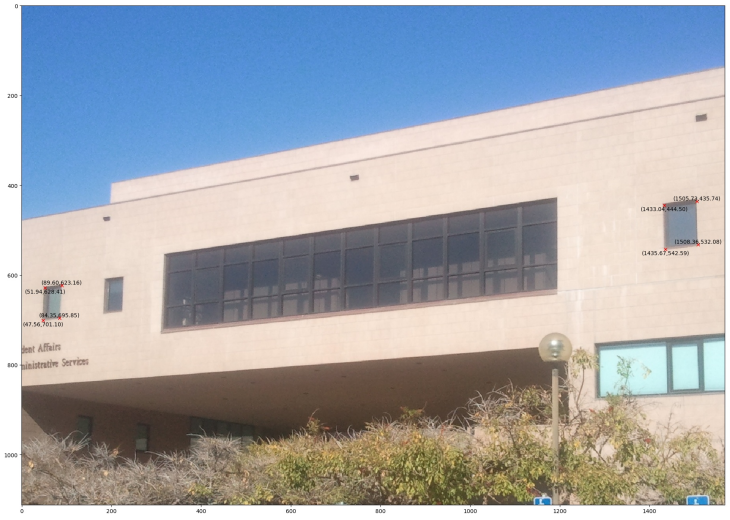

Figure 2: The coordinates are listed in table 2

Points	X	Y
Point 1	51.94	628.41
Point 2	47.56	701.10
Point 3	84.35	695.85
Point 4	89.60	623.16
Point 5	1433.04	444.5o
Point 6	1435.67	542.59
Point 7	1508.36	532.08
Point 8	1505.73	435.74

Table 2: For picture of school taken from the right

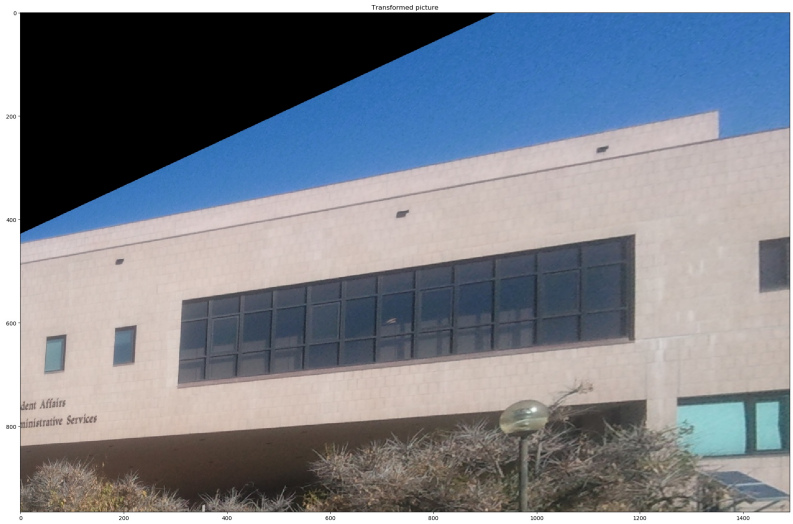

Figure 3: School now looks like it was taken from right

1.978	0.134	-28.543
0.706	1.523	-651.699
7.191e-0.4	4.096e-05	1

Table 3: Transformation matrix H from left angle to right

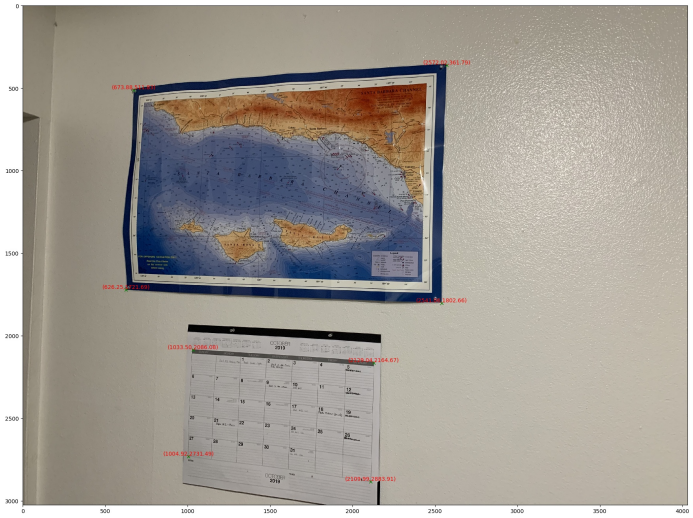

Figure 4: The coordinates are listed in table 4

Points	X	Y
Point 1	673.880	511.832
Point 2	626.248	1721.689
Point 3	2541.061	1802.664
Point 4	2572.022	361.790
Point 5	1033.503	2086.075
Point 6	1004.92	2731.491
Point 7	2109.99	2883.91
Point 8	2129.043	2164.668

Table 4: For picture of bed taken from the right

1.b Picture of bedroom wall

We did the same here. Here the coordinates are listed the same way, but points 5-8 is bottom poster, and 1-4 is top poster.

Doing the transformation on the picture taken from the left resulted in the picture in figure 6 These points resulted in the transformation matrix in table 6. For reference the code is listed in the last pages of this document.

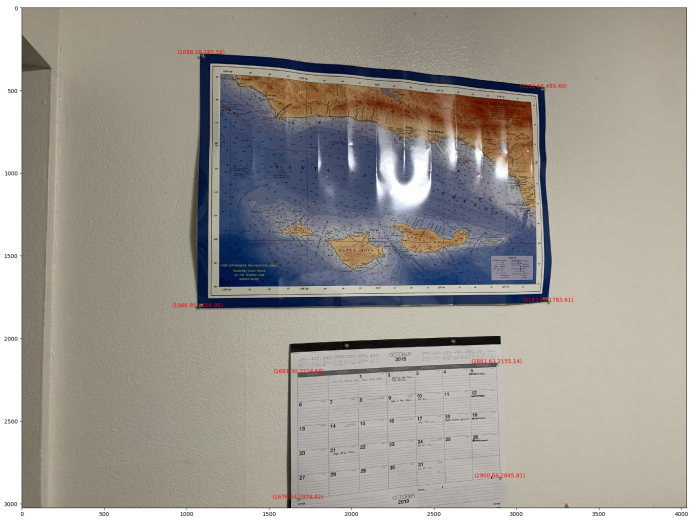

Figure 5: The coordinates are listed in table **??**

Points	X	Y
Point 1	1088.280	285.578
Point 2	1066.846	1816.953
Point 3	3193.62	1783.610
Point 4	3162.661	490.397
Point 5	1681.300	2214.681
Point 6	1676.538	2974.415
Point 7	2900.684	2845.808
Point 8	2881.631	2155.1416

Table 5: For picture of school taken from the left

0.4813	-1.97e-2	71.48
-0.1771	0.702	436.23
1.226e-0.4	7.61567e-06	1

Table 6: Transformation matrix H from left angle to right

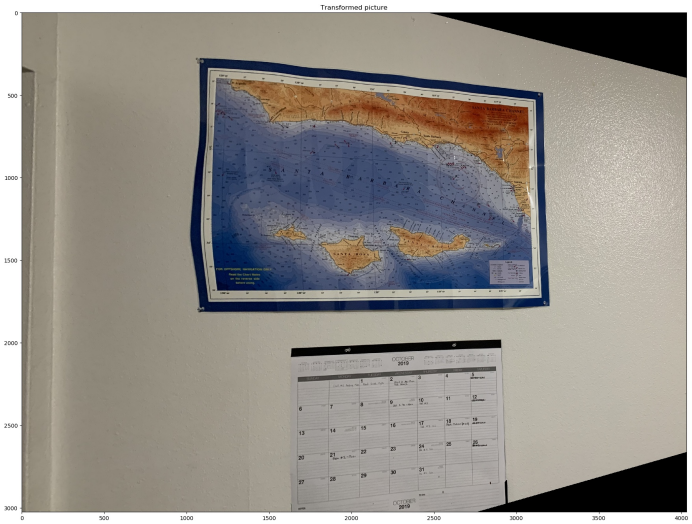

Figure 6: Wall now looks like it was taken from left

2 Code for homography transformation

```
#!/usr/bin/env python3

import helping_functions        as hf
import numpy                    as np
import  matplotlib.pyplot       as plt
from skimage import transform   as tf

def retrieveCorrespondingPoints(picturename1, picturename2):
    # INPUT: The filename of two pictures from different angles
    # OUTPUT: List with the pair of points in a tuple.
    # Output format: [((x1,y1), (x2,y2)), ((x11, y11), (x21, y21)), ...]
    print("You must choose 8 corresponding points"
          " in the same order for the both pictures")
    npoints = 8
    im1 = plt.imread(picturename1)

    plt.imshow(im1)
    picture1 = np.array(plt.ginput(npoints))


    im2 = plt.imread(picturename2)
    plt.imshow(im2)
    picture2 = np.array(plt.ginput(npoints))

    list_of_points = []
    for i in range(npoints):
        pointp1 = (picture1[i][0], picture1[i][1],)
        pointp2 = (picture2[i][0], picture2[i][1],)
        pair = (pointp2, pointp1,)
        list_of_points.append(pair)

    plt.close()
    ans = input('Would you like to save the points to a file? (y/n): ')
    if ans.lower() == 'y':
        filename = input("Insert filename (default: list): ")
        if filename != '':
            hf.save_list(list_of_points, "{}.txt".format(filename))
        else:
            hf.save_list(list_of_points)

    return list_of_points

def calculateHomographyTransformation(list_of_points):
    # INPUT: All the corresponding pairs of point
    # in the two pictures as a list with same index position
    # OUTPUT: Transformation matrix H
    # Here we want to find H that transforms the one picture to the other
    # transform from picture number 1 to picture number 2
    array_created = False
    for pair in list_of_points:
        x, y = pair[0]              # Picture 1
        x_prime, y_prime = pair[1]  # Picture 2

        if not array_created:
```

```
            A = np.array([[x, y, 1, 0, 0, 0, -x*x_prime, -y*x_prime],
                          [0, 0, 0, x, y, 1, -x*y_prime, -y*y_prime]])
            B = np.array([x_prime, y_prime])
            array_created = True

        else:
            A = np.append(A, np.array([[x, y, 1, 0, 0, 0, -x*x_prime, -y*x_prime],
                                       [0, 0, 0, x, y, 1, -x*y_prime, -y*y_prime]]),
                                       axis = 0)
            B = np.append(B, np.array([x_prime, y_prime]))

    # function return 8x1 with [h11 h12 h13 h21 h22 h23 h31 h32] here h33 = 1
    h = np.linalg.lstsq(A, B)[0]
    h = np.append(h, 1)

    H = np.reshape(h, (3, 3))
    return H

def transformImage(imagename, desiredImg, matrix = np.empty((3,3), dtype = float)):
    # INPUT: The image that is to be transformed,
    # the picture from the angle I should expect
    # and the transformation matrix
    transformation = tf.ProjectiveTransform(matrix)
    print(transformation)
    img = plt.imread(imagename)
    img2 = plt.imread(desiredImg)


    transformedImage = tf.warp(img, transformation)

    fig = plt.figure(0)
    sub1 = fig.add_subplot(1,1,1)
    sub1.set_title('Original Picture')
    plt.imshow(img)
    fig2 = plt.figure(1)
    sub2 = fig2.add_subplot(1,1,1)
    sub2.set_title('Transformed picture')
    plt.imshow(transformedImage)
    fig3 = plt.figure(2)
    sub3 = fig3.add_subplot(1,1,1)
    sub3.set_title('Desired picture')
    plt.imshow(img2)
    plt.show()
    return 0
```

3 Code helping functions used for save, load, etc

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt

def save_list(listname, filename = "list.txt"):
    with open(filename, 'w') as f:
        for item in listname:
            f.write("{}:{}\n".format(item[0], item[1]))

def load_list(filename = "list.txt"):
    list_of_points = []
    with open(filename, 'r') as f:
        for line in f:
            line = line.strip() # Remove whitespace \n etc
            p1, p2 = line.split(':')
            pointx1, pointy1 = (p1.strip('()')).split(',') # remove paranthesis
            pointx1 = float(pointx1)
            pointy1 = float(pointy1)

            pointx2, pointy2 = (p2.strip('()')).split(',') # Remove paranthesis
            pointx2 = float(pointx2)
            pointy2 = float(pointy2)

            list_of_points.append(((pointx1, pointy1,), (pointx2, pointy2,),))
    return list_of_points

def markPictures(image1, image2, list_of_points):
    list_of_points_image1 = []
    list_of_points_image2 = []

    for i in range(len(list_of_points)):
        list_of_points_image1.append(list_of_points[i][1])
        list_of_points_image2.append(list_of_points[i][0])

    list_of_x1 = []
    list_of_x2 = []
    list_of_y1 = []
    list_of_y2 = []

    for i in range(len(list_of_points_image1)):
        list_of_x1.append(list_of_points_image2[i][0])
        list_of_y1.append(list_of_points_image2[i][1])


    fig1, ax = plt.subplots()
    im1 = plt.imread(image2)
    plt.imshow(im1)
    for i in range(len(list_of_x1)):
        ax.plot([list_of_x1[i]], [list_of_y1[i]], 'gx')
        if i == 0:
            plt.text(list_of_x1[i], list_of_y1[i],
                "({ptx1:.2f},{pty1:.2f})".format(ptx1 = list_of_x1[i],
                                                 pty1 = list_of_y1[i]),
                ha = 'center', va = 'bottom',
                color = 'red',
```

```
                transform = ax.transData)
        elif i==1:
            plt.text(list_of_x1[i], list_of_y1[i],
                "({ptx1:.2f},{pty1:.2f})".format(ptx1 = list_of_x1[i],
                                                 pty1 = list_of_y1[i]),
                ha = 'center', va = 'bottom',
                color = 'red',
                transform = ax.transData)
        elif i==4:
            plt.text(list_of_x1[i], list_of_y1[i],
                "({ptx1:.2f},{pty1:.2f})".format(ptx1 = list_of_x1[i],
                                                 pty1 = list_of_y1[i]),
                ha = 'center', va = 'bottom',
                color = 'red',
                transform = ax.transData)
        elif i == 5:
            plt.text(list_of_x1[i], list_of_y1[i],
                "({ptx1:.2f},{pty1:.2f})".format(ptx1 = list_of_x1[i],
                                                 pty1 = list_of_y1[i]),
                ha = 'center', va = 'bottom',
                color = 'red',
                transform = ax.transData)
        else:
            plt.text(list_of_x1[i], list_of_y1[i],
                "({ptx1:.2f},{pty1:.2f})".format(ptx1 = list_of_x1[i],
                                                 pty1 = list_of_y1[i]),
                ha = 'center', va = 'bottom',
                color = 'red',
                transform = ax.transData)
    plt.show()
```