Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

Theory

How homography is calculated

We know that x' = Hx describes the point correspondences in a homography. In this homework, we want to find the homography between two images. We make use of point correspondences between images to determine the homography.

$$x' = Hx \Rightarrow \begin{bmatrix} x1' \\ x2' \\ x3' \end{bmatrix} = \begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & h33 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} h11x1 + h12x2 + h13x3 \\ h21x1 + h22x2 + h23x3 \\ h31x1 + h32x2 + h33x3 \end{bmatrix} \quad (1)$$

Since we are dealing with homogeneous coordinates, we are only interested in finding ratios, which is why we only want to find the first eight coordinates of the matrix H, setting h_{33} to 1. We will also set x3 as 1, so we can use the physical coordinates $x = \frac{x1}{x3}$ and $y = \frac{x2}{x3}$

The physical coordinates x' and y' can be calculated as:

$$x' = \frac{x1'}{x3} = \frac{h11x1+h12x2+h13x3}{h31x1+h32x2+h33x3} = \frac{h11x+h12y+h13}{h31x+h32y+1} \quad (2)$$

$$y' = \frac{x2'}{x3} = \frac{h21x1+h22x2+h23x3}{h31x1+h32x2+h33x3} = \frac{h21x+h22y+h23}{h31x+h32y+1} \quad (3)$$

Thus, for every pair of correspondence points between the source image and the target image, we get two equations. To get eight equations, we would need four correspondence points. Since we have eight unknowns in the matrix H, these eight equations will be enough to find the unknowns.

$$\begin{bmatrix} x1' \\ y1' \\ x2' \\ y2' \\ x3' \\ y3' \\ x4' \\ y4' \end{bmatrix} = \begin{bmatrix} x1 & y1 & 1 & 0 & 0 & 0 & -x1x1' & -y1x1' \\ 0 & 0 & 0 & x1 & y1 & 1 & -x1y1' & -y1y1' \\ x2 & y2 & 1 & 0 & 0 & 0 & -x2x2' & -y2x2' \\ 0 & 0 & 0 & x2 & y2 & 1 & -x2y2' & -y2y2' \\ x3 & y3 & 1 & 0 & 0 & 0 & -x3x3' & -y3x3' \\ 0 & 0 & 0 & x3 & y3 & 1 & -x3y3' & -y3y3' \\ x4 & y4 & 1 & 0 & 0 & 0 & -x4x4' & -y4x4' \\ 0 & 0 & 0 & x4 & y4 & 1 & -x4y4' & -y4y4' \end{bmatrix} \begin{bmatrix} h11 \\ h12 \\ h13 \\ h21 \\ h22 \\ h23 \\ h31 \\ h32 \end{bmatrix} \quad (4)$$

(4) is obtained by repeating (2) and (3) for four points. (4) can also be expressed as t = Ph. Premultiplying by P^{-1}, we get $h = P^{-1}t$. That is how we find the column vector h, which is then reshaped to obtain the 3x3 matrix H.

Tasks

Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

(a) (b)

(c) (d)

Figure 1: Images that were provided

Task 1(a)

A homography was found to transform Figure 1d to the PQRS polygon in Figure 1a. Then, a blacked-out dummy image of the size of Figure 1a is created, where all the pixels that lie in the quadrilateral to be replaced are whitened-out. Then, for all pixels of Figure 1a that lie in the quadrilateral to be replaced, the position of the pixel is post-multiplied with the homography matrix H to obtain the position from which the color values (red, green and blue) need to be fetched from Figure 1d. The position of the pixel which is obtained by multiplication of the matrices is not guaranteed to be an integer. In that case, the float value is rounded down to the nearest integer, and the color from that pixel is taken. In a more accurate implementation, interpolation to obtain colors from all neighboring pixels would be a good idea. The dummy image is useful in determining whether a particular pixel in Figure 1a needs to be replaced or not. Thus the quadrilateral outlined by PQRS in Figure 1a is replaced by a distorted version of Figure 1d. The positions of points P, Q, R and S were found using the software GIMP by hovering over the point with a mouse.

The same process is repeated to project Figure 1d on Figure 1b and Figure 1c. The results are shown in Figure 2.

Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

(a) (b)

(c)

Figure 2: Resulting images of Task 1(a)

Task 1(b)

Homographies are found between Figure 1a and 1b, and between Figure 1b and Figure 1c. The two matrices and multiplied. That product matrix is applied on Figure 1a to obtain a new image which resembles Figure 1c. The parts of Figure 1c which are not captured by the camera in Figure 1a are missing in the new image, as shown in Figure 3.

Rehana Mahfuz (rmahfuz@purdue.edu)

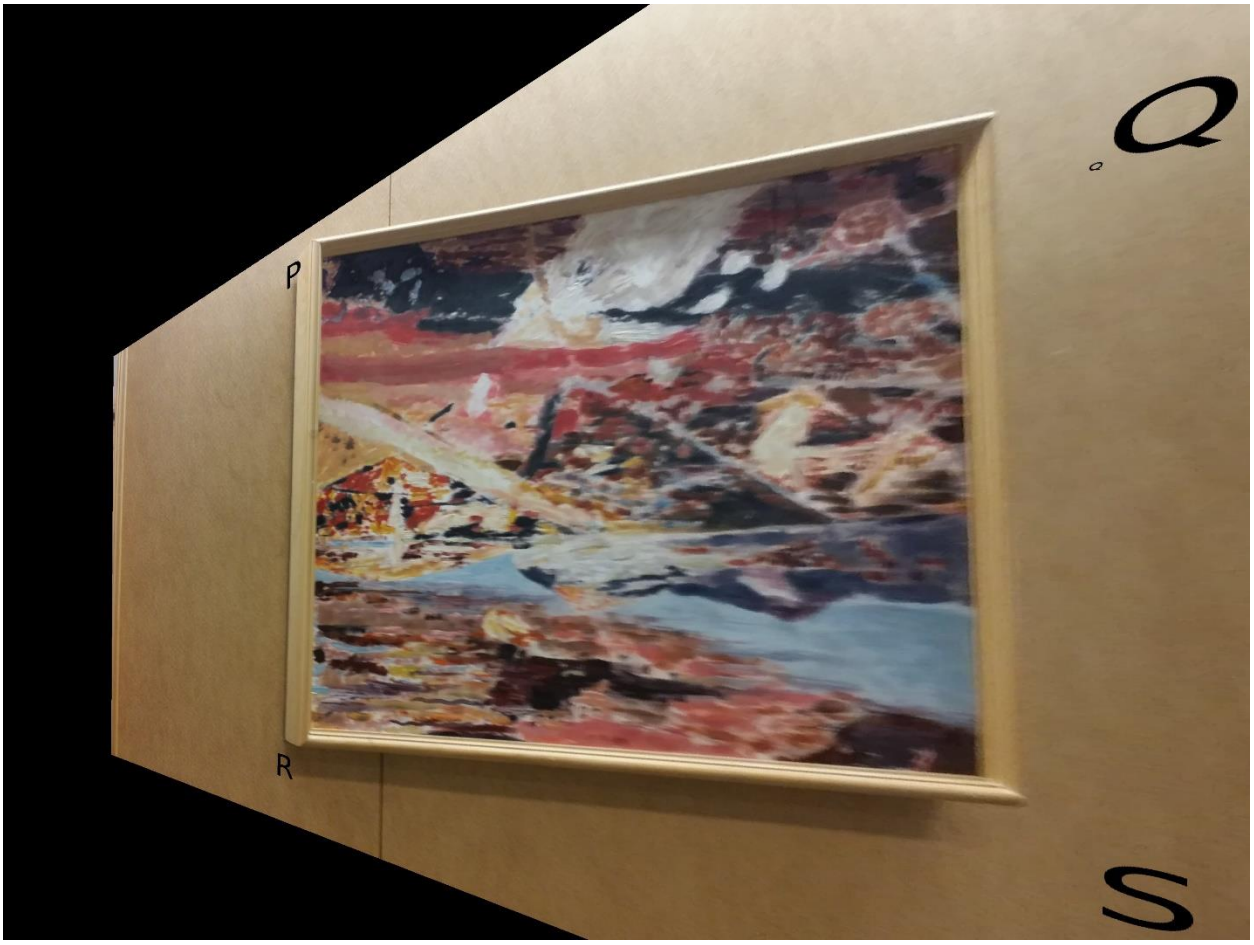


Figure 3: Result of Task 2(b)

Figure 5: Resulting images of Task 2

Task 2

Some images of a flyer were obtained from three different angles: left, center and right, as shown in Figure 4(a)-(c). I took a picture of myself, as shown in Figure 1(d), and projected my face on the flyer to replace the face of the Grand Prix Queen with my own.

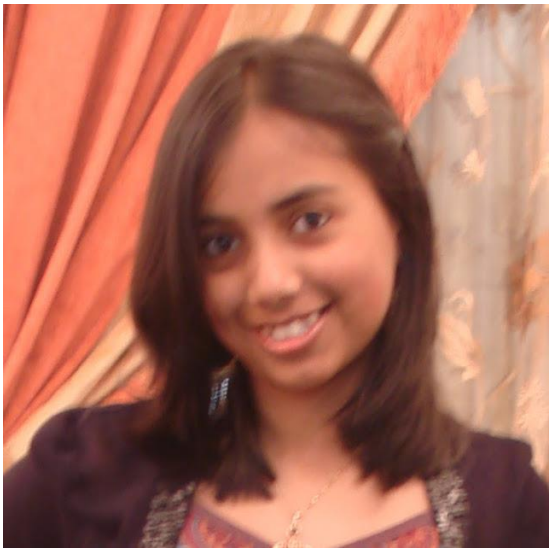
I followed the same procedure that I had followed to replace the picture in the frame with Jackie's face. I obtained the coordinates of the bounding polygons in Figures 4 (a)-(c) using GIMP. The contents of the polygons were replaced with my face. The results are shown in Figure 5.

Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

(a): Flyer from left

(b): Flyer from center

(c): Flyer from right

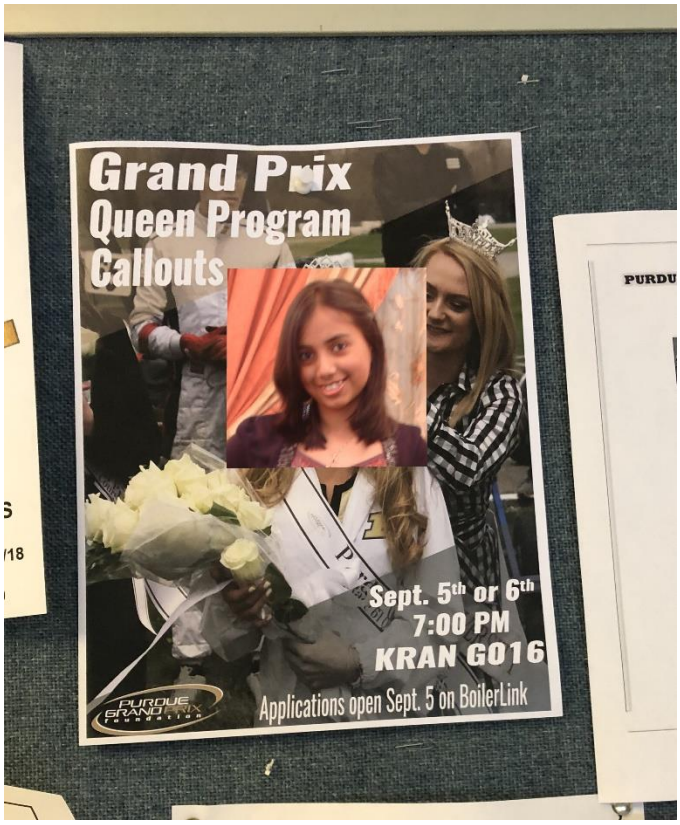


(d): A picture of me

Figure 4: Input images for Task 2

Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

(a) Projection of my face on flyer from left



(b) Projection of my face on flyer from the center

Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

(c) Projection of my face on flyer from the right

Code

```
import numpy as np

import cv2


#path = '../Users/rmahfuz/Desktop/661/HW02/'

path = ''

#================================================================================================

def find_homography(x, x_dash):

    '''x and x_dash are lists of four lists, each list containing two coordinates: [x,y]

    returns H = inv(P)*t'''

    t = np.array(x_dash, dtype = float).flatten().reshape((8,1))
```

```
    P = []
    for i in range(4):
        P.append([x[i][0], x[i][1], 1, 0, 0, 0, -1*x[i][0]*x_dash[i][0], -1*x[i][1]*x_dash[i][0]])
        P.append([0, 0, 0, x[i][0], x[i][1], 1, -1*x[i][0]*x_dash[i][1], -1*x[i][1]*x_dash[i][1]])
    P = np.array(P, dtype = float)#; print(P)
    P_inv = np.linalg.inv(P)
    H = np.matmul(P_inv, t) # H = inv(P)*t
    return H
#-------------------------------------------------------------------------------------------------
def apply_homography(im1, im2, pts1, pts2):
    '''img1 is destination img, img2 is source image. Returns im1 with distorted im2 in it'''
    H = find_homography(pts1, pts2)
    H = np.insert(H, 8, 1).reshape((3,3))
    return find_result(im1, im2, pts1, H)
#-------------------------------------------------------------------------------------------------
def find_result(im1, im2, pts1, H):
    '''Returns the resulting image'''
    #creating the 'dummy' image which is completely blacked out except at the pixels which need to be replaced
    dummy = np.zeros((im1.shape[0],im1.shape[1],3),dtype='uint8') #completely blacked out
    pts = np.array([[pts1[0][1], pts1[0][0]], [pts1[1][1], pts1[1][0]], [pts1[2][1], pts1[2][0]], [pts1[3][1], pts1[3][0]]], np.int32) #pixels that need to be whitened
    pts = pts.reshape((-1,1,2))
    cv2.fillPoly(dummy,[pts],(255,255,255)) #whitening those pixels
    cv2.imwrite(path + 'dummy.jpg', dummy)
    #-------------------------------------------------------------------------------------------------
    #Filling im1 with distorted im2:
    for i in range(im1.shape[0]): #till 2709
        for j in range(im1.shape[1]): #till 3612
            if dummy[i][j][0] == 255: #change the contents
                source = np.matmul(H, [[i], [j], [1]]);#print(source, '\n')
```

```
            source /= source[2][0]#; print(source)

            if source[0][0] > 0 and source[1][0] > 0 and source[0][0] < im2.shape[0] and source[1][0] <
im2.shape[1]:

                im1[i][j] = im2[int(source[0][0]), int(source[1][0])]

    return im1

#================================================================================
==================


if __name__ == '__main__':

    #Reading the images

    img1 = cv2.imread(path + '1.jpg')

    img2 = cv2.imread(path + '2.jpg')

    img3 = cv2.imread(path + '3.jpg')

    img4 = cv2.imread(path + 'Jackie.jpg')


    #coordinates found using GIMP:

    img1PQRS = np.array([[165,1501], [720,2970], [2046,3008], [2223,1485]], dtype = float)

    img2PQRS = np.array([[332,1320], [608,3014], [1900,3030], [2012,1300]], dtype = float)

    img3PQRS = np.array([[736,918], [390,2810], [2232,2846], [2084,891]], dtype = float)

    img4PQRS = np.array([[0   ,0  ], [0,1280  ], [720,1280 ], [720,0 ]], dtype = float)

    #------------------------------Part a---------------------------------------------------------------------

    #Projecting Jackie on 1a

    cv2.imwrite(path + 'result1.jpg', apply_homography(img1, img4, img1PQRS, img4PQRS))

    #Projecting Jackie on 1b

    cv2.imwrite(path + 'result2.jpg', apply_homography(img2, img4, img2PQRS, img4PQRS))

    #Projecting Jackie on 1c

    cv2.imwrite(path + 'result3.jpg', apply_homography(img3, img4, img3PQRS, img4PQRS))

    #------------------------------Part b---------------------------------------------------------------------

    Hab = find_homography(img2PQRS, img1PQRS)

    Hab = np.insert(Hab, 8, 1).reshape((3,3)); print(Hab)

    Hbc = find_homography(img3PQRS, img2PQRS)
```

Rehana Mahfuz (rmahfuz@purdue.edu)
ECE 661 HW02

```
Hbc = np.insert(Hbc, 8, 1).reshape((3,3)) ; print(Hbc); print(np.matmul(Hab, Hbc))
blank = np.zeros((img1.shape[0],img1.shape[1],3),dtype='uint8') #blank image
endPQRS = np.array([[0   ,0  ], [0,3612  ], [2709,3612 ], [2709,0 ]], dtype = float) #ends of the image
img1 = cv2.imread(path + '1.jpg')
cv2.imwrite(path + 'part_b.jpg', find_result(blank, img1, endPQRS, np.matmul(Hab, Hbc)))
#----------------------------Part c-------------------------------------------------------------------------
straightPQRS = np.array([[948,800  ], [944,1512 ], [1656,1516], [1664,796 ]], dtype = float)
leftPQRS     = np.array([[700,460   ], [532,1036 ], [1680,1084], [1548,488]], dtype = float)
rightPQRS    = np.array([[1122,1562], [1303,1991], [2216,1947], [2304,1402]], dtype = float)
mePQRS       = np.array([[0   ,0   ], [0,640    ], [637,640  ], [637,0    ]], dtype = float)

straight = cv2.imread(path + 'straight.jpeg')
left     = cv2.imread(path + 'left.jpeg'    )
right    = cv2.imread(path + 'right.jpeg'   )
me       = cv2.imread(path + 'me.jpg'   )

cv2.imwrite(path + 'me_straight.jpg', apply_homography(straight, me, straightPQRS, mePQRS))
cv2.imwrite(path + 'me_left.jpg', apply_homography(left, me, leftPQRS, mePQRS))
cv2.imwrite(path + 'me_right.jpg', apply_homography(right, me, rightPQRS, mePQRS))
```