COMPUTER VISION
ECE:66100

Homework Assignment 2

Name: Amruthavarshini Talikoti
Email ID: atalikot@purdue.edu
Student ID: 0029949176
Purdue University
Department of Electrical and Computer Engineering

Task 1

Task 1a

Homography Computation

This task comprises of two steps- first, finding the homography between pair of images (1){1(a) and 1(d)}, (2){1(b) and 1(d)} and (3){1(c) and 1(d)} and second, finding the mapping from frame plane to the personality plane and determining the weighted pixel value at that location to be placed in the frame picture.
Let the homography matrix be denoted by H that transforms point with homogeneous coordinates $(x, y, 1)$ in the frame plane to the point with homogeneous coordinates $(x', y', 1)$ in the personality frame. So,

$$Let \quad H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

The element of $H_{(3,3)}$ is taken to be 1, since we are talking about homogeneous coordinates while considering homographies and ratios are all that matter. Now, the transformation of points from frame plane to personality frame by homography H is formulated as follows,

$$H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

These lead to the following set of equations,

$$ax + by + c = x'$$
$$dx + ey + f = y'$$
$$gx + hy + 1 = 1$$

Thus, the coordinates in the personality frame can be rearranged as,

$$x' = \frac{ax + by + c}{gx + hy + 1}$$
$$y' = \frac{dx + ey + f}{gx + hy + 1}$$

The equations can be rearranged as follows,

$$x' = ax + by + c - gxx' - hyx'$$
$$y' = dx + ey + f - gxy' - hyy'$$

These equations contain 8 unknowns and 2 equations. So, if four such point transformations are considered, then we can obtain a system of 8 unknowns with 8 equations that can be solved. Consider points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ and (x_4, y_4) mapped to $(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3)$ and (x'_4, y'_4) respectively. This system of equation can be represented using matrices as follows.

$$AH = b$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$$

Thus, the homography matrix can be calculated as $H = A^{-1}b$ from the known A and b matrices and rearranged to its standard 3X3 matrix form as,

$$H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Weighted mapping of pixels

Now, traverse through the coordinates of the frame picture and for each coordinate, find the transformation using H that corresponds to the coordinates in the personality frame. Let this coordinate that is mapped to be p. If this pair of coordinates lies outside the dimensions of the personality picture, use the pixel value of the frame picture as the required pixel of the result framed picture. Otherwise, for the coordinates being mapped to the area of the personality picture, choose the nearest 4 neighbours with integer coordinates surrounding the coordinate to which H mapped the frame coordinates. Let these four neighbors be n_1, n_2, n_3 and n_4. So, for the pixel value of p, a weighted average of the pixel values at the integer coordinate neighbors n_1, n_2, n_3 and n_4 is taken. The weights are calculated using the following formula,

$$w_i = \frac{1}{distance\ \ between\ \ p\ \ and\ \ n_i}$$

Thus, the weighted average for the pixel value at p is taken as

$$p = \frac{w_1 p_{n_1} + w_2 p_{n_2} + w_3 p_{n_3} + w_4 p_{n_4}}{w_1 + w_+ w_3 + w_4}$$

(wherein $p_{n_1}, p_{n_2}, p_{n_3}$ and p_{n_4} are the pixel values at n_1, n_2, n_3 and n_4 respectively) is assigned as the required pixel of the result framed picture.
A similar procedure is carried out for pictures (2){1(b) and 1(d)} and (3){1(c) and 1(d)}.

Figure 1: The first frame plane with coordinates P, Q, R and S for Task 1a

Figure 2: The second frame plane with coordinates P, Q, R and S for Task 1a

Figure 3: The third frame plane with coordinates P, Q, R and S for Task 1a

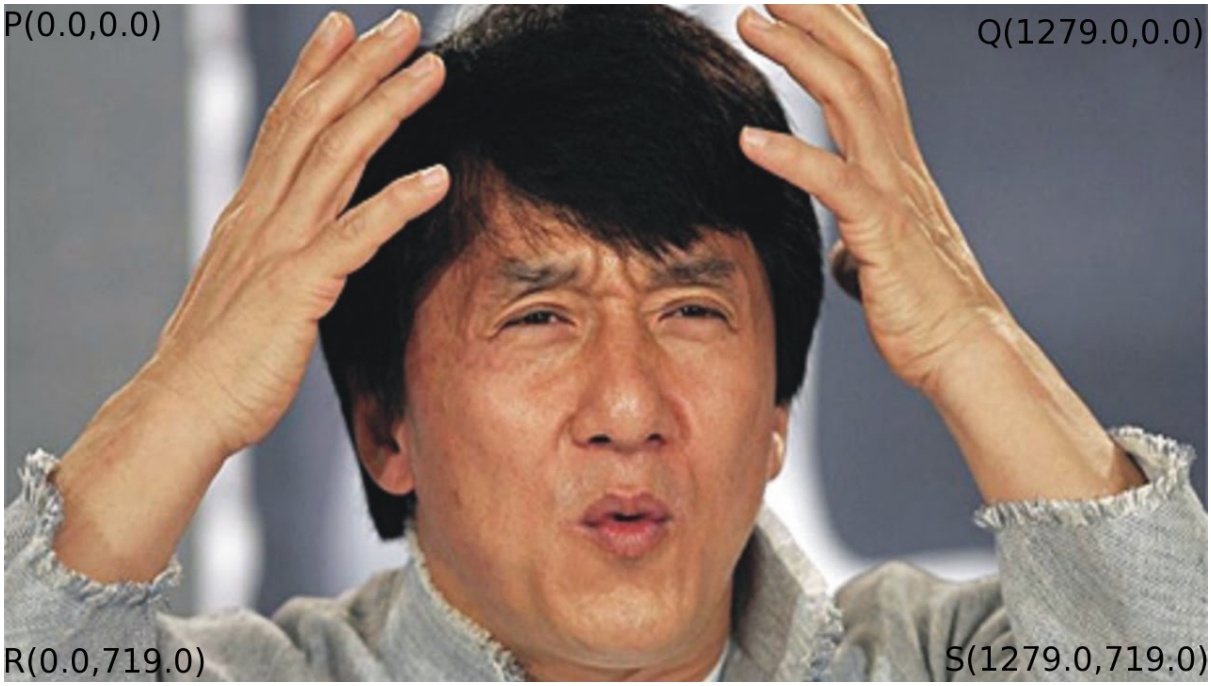

Figure 4: The personality plane with coordinates P, Q, R and S for Task 1a

Figure 5: The projection of the personality on the first frame plane with coordinates P, Q, R and S for Task 1a

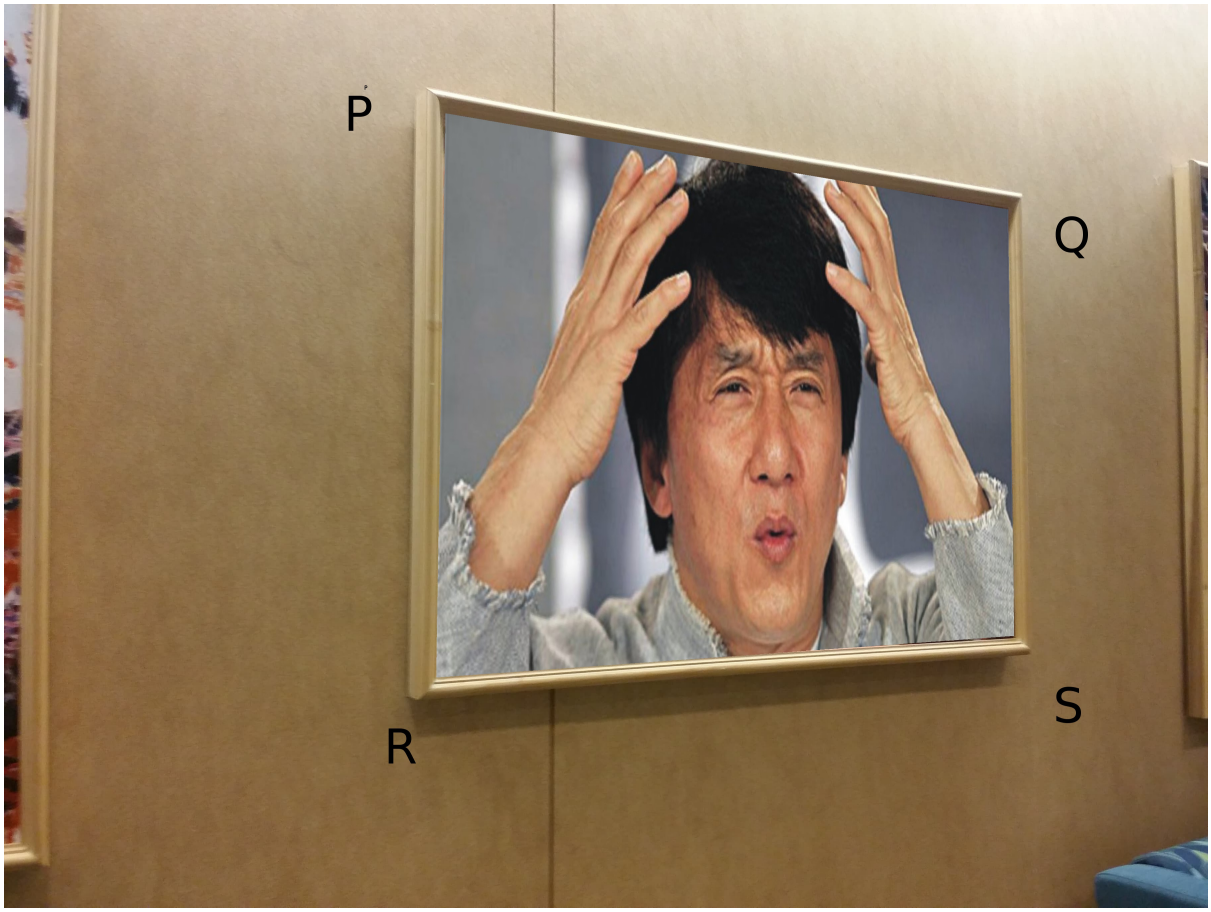

Figure 6: The projection of the personality on the second frame plane with coordinates P, Q, R and S for Task 1a

Figure 7: The projection of the personality on the third frame plane with coordinates P, Q, R and S for Task 1a

Task 1b

Computations similar to the ones described in the previous section are used to determine the homographies between images {1a and 1b} and {1b and 1c}. Let the homographies so obtained be referred to as H_{ab} and H_{bc} respectively. The product of these two matrices results in the homography (since the homographies form a group with matrix multiplication as the binary operator) $H_{ac} = H_{ab} * H_{bc}$ between images {1a and 1c}.
Further, each of the coordinates in the result picture (intended to be in the orientation as 1c) are multiplied with H_{ac}^{-1} to find the corresponding coordinates in the frame plane (1a). If these mapped coordinates lie outside the dimensions of picture (1a) then the pixel value is taken as RGB=[255 255 255] or = [0 0 0], i.e., just a white or black pixel. Otherwise, a procedure similar to the one described in previous subsection is carried out to find the 4 nearest integer coordinate neighbors and the weighted average of the pixel values of these 4 gives the

required value of the pixel in the result picture. This ultimately results in the picture of image (1a) being displayed in an orientation similar to (1c).

Figure 8: The resultant image when the product homography H_{ac} is applied to first frame plane with coordinates P, Q, R and S for Task 1b

Task 2

The task involves repeating the entire procedure in Task 1 with new source images. A similar logic flow and code was used as described above. The obtained resultant images are as shown below.

Figure 9: The first frame plane with coordinates P, Q, R and S for Task 2

Figure 10: The second frame plane with coordinates P, Q, R and S for Task 2

Figure 11: The third frame plane with coordinates P, Q, R and S for Task 2

Figure 12: The personality plane with coordinates P, Q, R and S for Task 2

Figure 13: The projection of the personality on the first frame plane with coordinates P, Q, R and S for Task 2

Figure 14: The projection of the personality on the second frame plane with coordinates P, Q, R and S for Task 2

Figure 15: The projection of the personality on the third frame plane with coordinates P, Q, R and S for Task 2

Figure 16: The resultant image when the product homography H_{ac} is applied to first frame plane with coordinates P, Q, R and S for Task 2

Source Code

Task 1

```
import cv2 as cv
import math
import numpy as np

# TASK 1-A

# Loading the images
frame=cv.imread("1.jpg")
person=cv.imread("Jackie.jpg")

# The four coordinates of the frame
F=list()
F.append((1468.0,60.0,1.0))
F.append((3028.0,676.0,1.0))
F.append((1432.0,2332.0,1.0))
F.append((3040.0,2080.0,1.0))

# The four coordinates of personality
P=[]
P.append((0.0,0.0,1.0))
P.append((1279.0,0.0,1.0))
P.append((0.0,719.0,1.0))
P.append((1279.0,719.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
```

```
        A[2*i+0][4]=0.0
        A[2*i+0][5]=0.0
        A[2*i+0][6]=-1*F[i][0]*P[i][0]
        A[2*i+0][7]=-1*F[i][1]*P[i][0]
        A[2*i+1][0]=0.0
        A[2*i+1][1]=0.0
        A[2*i+1][2]=0.0
        A[2*i+1][3]=F[i][0]
        A[2*i+1][4]=F[i][1]
        A[2*i+1][5]=1.0
        A[2*i+1][6]=-1*F[i][0]*P[i][1]
        A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
        b[2*i+0][0]=P[i][0]
        b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
```

```
H[2][2]=1.0

# Traversing through pixels and transforming it using H.
# Retaining old value if out of person boundary else
# Taking weighted value of pixel in person frame
framed=np.zeros((frame.shape[0],frame.shape[1],3))
for x in range(frame.shape[0]):
    for y in range(frame.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>719 or p_coord[0]<0 or p_coord[0]>1279:
            framed[x][y]=frame[x][y]
        else:
            n1=person[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=person[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=person[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=person[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_1.jpg",framed)



# Loading the images
frame=cv.imread("2.jpg")
person=cv.imread("Jackie.jpg")

# The four coordinates of the frame
F=list()
```

```
F.append((1324.0,327.0,1.0))
F.append((3012.0,615.0,1.0))
F.append((1296.0,2018.0,1.0))
F.append((3030.0,1893.0,1.0))


# The four coordinates of personality
P=[]
P.append((0.0,0.0,1.0))
P.append((1279.0,0.0,1.0))
P.append((0.0,719.0,1.0))
P.append((1279.0,719.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)
```

```
# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

# Traversing through pixels and transforming it using H.
# Retaining old value if out of person boundary else
# Taking weighted value of pixel in person frame
framed=np.zeros((frame.shape[0],frame.shape[1],3))
for x in range(frame.shape[0]):
    for y in range(frame.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>719 or p_coord[0]<0 or p_coord[0]>1279:
            framed[x][y]=frame[x][y]
        else:
```

```
            n1=person[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=person[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=person[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=person[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_2.jpg",framed)

# Loading the images
frame=cv.imread("3.jpg")
person=cv.imread("Jackie.jpg")

# The four coordinates of the frame
F=list()
F.append((921.0,730.0,1.0))
F.append((2803.0,381.0,1.0))
F.append((900.0,2096.0,1.0))
F.append((2845.0,2227.0,1.0))


# The four coordinates of personality
P=[]
P.append((0.0,0.0,1.0))
P.append((1279.0,0.0,1.0))
P.append((0.0,719.0,1.0))
P.append((1279.0,719.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
```

```
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
```

```
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

# Traversing through pixels and transforming it using H.
# Retaining old value if out of person boundary else
# Taking weighted value of pixel in person frame
framed=np.zeros((frame.shape[0],frame.shape[1],3))
for x in range(frame.shape[0]):
    for y in range(frame.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>719 or p_coord[0]<0 or p_coord[0]>1279:
            framed[x][y]=frame[x][y]
        else:
            n1=person[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=person[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=person[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=person[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_3.jpg",framed)


# TASK 1-B

# The four coordinates of the frame 1a
F=list()
```

```
F.append((1468.0,60.0,1.0))
F.append((3028.0,676.0,1.0))
F.append((1432.0,2332.0,1.0))
F.append((3040.0,2080.0,1.0))

# The four coordinates of the frame 1b
P=[]
P.append((1324.0,327.0,1.0))
P.append((3012.0,615.0,1.0))
P.append((1296.0,2018.0,1.0))
P.append((3030.0,1893.0,1.0))


# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)
```

```
# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

H_ab=H


# The four coordinates of the frame 1b
F=list()
F.append((1324.0,327.0,1.0))
F.append((3012.0,615.0,1.0))
F.append((1296.0,2018.0,1.0))
F.append((3030.0,1893.0,1.0))

# The four coordinates of the frame 1c
P=[]
P.append((921.0,730.0,1.0))
```

```
P.append((2803.0,381.0,1.0))
P.append((900.0,2096.0,1.0))
P.append((2845.0,2227.0,1.0))


# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
```

```
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

H_bc=H

# Calculating homography between a and c
H_ac=np.matmul(H_ab,H_bc)
H=H_ac
H_inv=np.linalg.pinv(H)

# Applying product homography to 1a

# Loading the images
frame_sh=cv.imread("3.jpg")
frame=cv.imread("1.jpg")
# Traversing through pixels and transforming it using H
framed=np.zeros((frame_sh.shape[0],frame_sh.shape[1],3))
for x in range(frame_sh.shape[0]):
    for y in range(frame_sh.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H_inv,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>(frame.shape[0]-1) or p_coord[0]<0 or p_coord
```

```
            framed[x][y]=[255.0,255.0,255.0]
        else:
            n1=frame[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=frame[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=frame[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=frame[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_ac.jpg",framed)
```

Task 2

```
import cv2 as cv
import math
import numpy as np

# TASK 1-A

# Loading the images
frame=cv.imread("1_a.jpg")
person=cv.imread("hima.jpg")

# The four coordinates of the frame
F=list()
F.append((1944.0,408.0,1.0))
F.append((3812.0,756.0,1.0))
F.append((1956.0,2964.0,1.0))
F.append((3804.0,2592.0,1.0))

# The four coordinates of personality
P=[]
P.append((0.0,0.0,1.0))
```

```
P.append((669.0,0.0,1.0))
P.append((0.0,736.0,1.0))
P.append((669.0,736.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)
```

```
# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

# Traversing through pixels and transforming it using H.
# Retaining old value if out of person boundary else
# Taking weighted value of pixel in person frame
framed=np.zeros((frame.shape[0],frame.shape[1],3))
for x in range(frame.shape[0]):
    for y in range(frame.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>(person.shape[0]-1) or p_coord[0]<0 or p_coor
            framed[x][y]=frame[x][y]
        else:
            n1=person[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=person[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=person[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=person[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)
```

```
cv.imwrite("framed_1_b.jpg",framed)



# Loading the images
frame=cv.imread("1_b.jpg")
person=cv.imread("hima.jpg")

# The four coordinates of the frame
F=list()
F.append((1088.0,440.0,1.0))
F.append((3828.0,452.0,1.0))
F.append((1088.0,2888.0,1.0))
F.append((3824.0,2880.0,1.0))



# The four coordinates of personality
P=[]
P.append((0.0,0.0,1.0))
P.append((669.0,0.0,1.0))
P.append((0.0,736.0,1.0))
P.append((669.0,736.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
```

```
        A[2*i+1][0]=0.0
        A[2*i+1][1]=0.0
        A[2*i+1][2]=0.0
        A[2*i+1][3]=F[i][0]
        A[2*i+1][4]=F[i][1]
        A[2*i+1][5]=1.0
        A[2*i+1][6]=-1*F[i][0]*P[i][1]
        A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
        b[2*i+0][0]=P[i][0]
        b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

# Traversing through pixels and transforming it using H.
# Retaining old value if out of person boundary else
```

```
# Taking weighted value of pixel in person frame
framed=np.zeros((frame.shape[0],frame.shape[1],3))
for x in range(frame.shape[0]):
    for y in range(frame.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>(person.shape[0]-1) or p_coord[0]<0 or p_coor
            framed[x][y]=frame[x][y]
        else:
            n1=person[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=person[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=person[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=person[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_2_b.jpg",framed)

# Loading the images
frame=cv.imread("1_c.jpg")
person=cv.imread("hima.jpg")

# The four coordinates of the frame
F=list()
F.append((796.0,640.0,1.0))
F.append((3064.0,224.0,1.0))
F.append((808.0,2688.0,1.0))
F.append((3100.0,2984.0,1.0))
```

```
# The four coordinates of personality
P=[]
P.append((0.0,0.0,1.0))
P.append((669.0,0.0,1.0))
P.append((0.0,736.0,1.0))
P.append((669.0,736.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]
```

```
# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

# Traversing through pixels and transforming it using H.
# Retaining old value if out of person boundary else
# Taking weighted value of pixel in person frame
framed=np.zeros((frame.shape[0],frame.shape[1],3))
for x in range(frame.shape[0]):
    for y in range(frame.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>(person.shape[0]-1) or p_coord[0]<0 or p_coor
            framed[x][y]=frame[x][y]
        else:
            n1=person[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=person[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=person[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=person[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
```

```
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_3_b.jpg",framed)



# TASK 1-B

# The four coordinates of the frame 1a
F=list()
F.append((1944.0,408.0,1.0))
F.append((3812.0,756.0,1.0))
F.append((1956.0,2964.0,1.0))
F.append((3804.0,2592.0,1.0))

# The four coordinates of the frame 1b
P=[]
P.append((1088.0,440.0,1.0))
P.append((3828.0,452.0,1.0))
P.append((1088.0,2888.0,1.0))
P.append((3824.0,2880.0,1.0))


# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
```

```
        A[2*i+0][7]=-1*F[i][1]*P[i][0]
        A[2*i+1][0]=0.0
        A[2*i+1][1]=0.0
        A[2*i+1][2]=0.0
        A[2*i+1][3]=F[i][0]
        A[2*i+1][4]=F[i][1]
        A[2*i+1][5]=1.0
        A[2*i+1][6]=-1*F[i][0]*P[i][1]
        A[2*i+1][7]=-1*F[i][1]*P[i][1]

# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
        b[2*i+0][0]=P[i][0]
        b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

H_ab=H
```

```
# The four coordinates of the frame 1b
F=list()
F.append((1088.0,440.0,1.0))
F.append((3828.0,452.0,1.0))
F.append((1088.0,2888.0,1.0))
F.append((3824.0,2880.0,1.0))

# The four coordinates of the frame 1c
P=[]
P.append((796.0,640.0,1.0))
P.append((3064.0,224.0,1.0))
P.append((808.0,2688.0,1.0))
P.append((3100.0,2984.0,1.0))

# Constructing the matrix for 8 equations with 8 unknowns
A=np.zeros((8,8))
for i in range(4):
    A[2*i+0][0]=F[i][0]
    A[2*i+0][1]=F[i][1]
    A[2*i+0][2]=1.0
    A[2*i+0][3]=0.0
    A[2*i+0][4]=0.0
    A[2*i+0][5]=0.0
    A[2*i+0][6]=-1*F[i][0]*P[i][0]
    A[2*i+0][7]=-1*F[i][1]*P[i][0]
    A[2*i+1][0]=0.0
    A[2*i+1][1]=0.0
    A[2*i+1][2]=0.0
    A[2*i+1][3]=F[i][0]
    A[2*i+1][4]=F[i][1]
    A[2*i+1][5]=1.0
    A[2*i+1][6]=-1*F[i][0]*P[i][1]
    A[2*i+1][7]=-1*F[i][1]*P[i][1]
```

```
# The inverse of A
A_I=np.linalg.pinv(A)

# Constructing the 8X1 b matrix
b=np.zeros((8,1))
for i in range(4):
    b[2*i+0][0]=P[i][0]
    b[2*i+1][0]=P[i][1]

# Evaluating the homograph matrix
H_col=np.matmul(A_I,b)

# Converting the H to a 3X3 matrix
H=np.zeros((3,3))
H[0][0]=H_col[0]
H[0][1]=H_col[1]
H[0][2]=H_col[2]
H[1][0]=H_col[3]
H[1][1]=H_col[4]
H[1][2]=H_col[5]
H[2][0]=H_col[6]
H[2][1]=H_col[7]
H[2][2]=1.0

H_bc=H

# Calculating homography between a and c
H_ac=np.matmul(H_ab,H_bc)
H=H_ac
H_inv=np.linalg.pinv(H)

# Applying product homography to 1a

# Loading the images
```

```
frame_sh=cv.imread("1_c.jpg")
frame=cv.imread("1_a.jpg")
# Traversing through pixels and transforming it using H
framed=np.zeros((frame_sh.shape[0],frame_sh.shape[1],3))
for x in range(frame_sh.shape[0]):
    for y in range(frame_sh.shape[1]):
        f_coord=(float(y),float(x),1.0)
        p_coord=np.matmul(H_inv,f_coord)
        factor=p_coord[2]
        p_coord=p_coord/factor
        if p_coord[1]<0 or p_coord[1]>(frame_sh.shape[0]-1) or p_coord[0]<0 or p_co
            framed[x][y]=[0.0,0.0,0.0]
        else:
            n1=frame[int(np.floor(p_coord[1]))][int(np.floor(p_coord[0]))]
            n2=frame[int(np.floor(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n3=frame[int(np.ceil(p_coord[1]))][int(np.ceil(p_coord[0]))]
            n4=frame[int(np.ceil(p_coord[1]))][int(np.floor(p_coord[0]))]
            w1=1/np.linalg.norm(n1-p_coord)
            w2=1/np.linalg.norm(n2-p_coord)
            w3=1/np.linalg.norm(n3-p_coord)
            w4=1/np.linalg.norm(n4-p_coord)
            framed[x][y]=((w1*n1)+(w2*n2)+(w3*n3)+(w4*n4))/(w1+w2+w3+w4)

cv.imwrite("framed_ac_b.jpg",framed)
```