1. ***You are given four images which are shown in Figs. 1a, 1b, 1c and 1d. The first three images consist of a painting hanging on a wall, while the fourth image contains the face of a famous celebrity. Complete the following tasks using the images.***

(a) (b) (c) (d)

Figure 1: Sample Images

(a) ***Project the face shown in Fig. 1d on the frame PQRS shown in Figs. 1a, 1b, 1c. For this task you need to find the homographies between the following pairs of images: (1) images shown in Figs. 1d and 1a (2) (1) images shown in Figs. 1d and 1b (3) (1) images shown in Figs. 1d and 1c***

Solution: The planar projective transformation of a point $\vec{X}$ in the homogeneous representational space is given by,

$$\vec{X}' = H\vec{X}, \tag{1}$$

where $\vec{X}'$ is the transformation of $\vec{X}$ and H is a non-singular 3×3 matrix and is called the homography between $\vec{X}'$ and $\vec{X}$. In general, H can be represented as,

$$H = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & \nu \end{bmatrix}. \tag{2}$$

We can set $\nu = 1$, since in the representational space, it is not the absolute values that matter but the ratios. If

$$\vec{X}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \vec{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix},$$

where (x, y) and (x', y') are the points in the physical space then Eq. 1 can be written as,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3}$$

Expanding Eq. 3 we get,

$$\begin{aligned} x' &= a_{11}x + a_{12}y + t_x \\ y' &= a_{21}x + a_{22}y + t_y \\ 1 &= v_1x + v_2y + 1, \end{aligned} \tag{4}$$

or,

$$x' = \frac{a_{11}x + a_{12}y + t_x}{v_1x + v_2y + 1}$$
$$y' = \frac{a_{21}x + a_{22}y + t_y}{v_1x + v_2y + 1}. \tag{5}$$

Given the pair of points, (x, y) and (x', y'), using Eq. 4 we can determine the unknown parameters of H namely, $a_{11}, a_{12}, a_{21}, a_{22}, t_x, t_y, v_1, v_2$. Since there are eight unknown parameters we need at least 8 equations. Each pair of (x, y), (x', y') points gives two equations. Hence for eight equations we will need four pairs of (x_i, y_i), (x'_i, y'_i) points where $i = 1, 2, 3, 4$. Substituting the values of (x_i, y_i), (x'_i, y'_i) in Eq. 4 we get a set of equations which can be written as,

$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \\ v_x \\ v_y \end{bmatrix} \tag{6}$$

or,

$$C' = AB, \tag{7}$$

where,

$$C' = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} B = \begin{bmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \\ v_x \\ v_y \end{bmatrix}.$$

We also define C as,

$$C = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix}, \tag{8}$$

for coding purposes.

For each calculation of H between Figs. 1a, 1b, 1c and Fig. 1d we use the four pairs of points PP′, QQ′, RR′ and SS′ where P, Q, R and S correspond to the bounding box of the frame in Figs. 1a, 1b, 1c and P′, Q′, R′ and S′ are the points corresponding to the four corners of Fig 1d. The coordinates of points P, Q, R, S and P′, Q′, R′ and S′ were obtained using IrfanView and are given below:

Substituting the values of (x,y) and (x', y') into Eq. 7 we get $a_{11}, a_{12}, a_{21}, a_{22}, t_x, t_y, v_1, v_2$.

Table 1: X and Y coordinates of points P, Q, R, S and P′, Q′, R′ S′

Fig. 1a.

	x_i	y_i
P	1500	112
Q	2962	730
R	1485	2243
S	3015	2054

Fig. 1b.

	x_i	y_i
P	1323	321
Q	3015	598
R	1299	2007
S	3033	1898

Fig. 1c.

	x_i	y_i
P	916	725
Q	2809	371
R	890	2080
S	2847	2231

Fig. 1d.

	x'_i	y'_i
P′	0	0
Q′	1279	0
R′	0	719
S′	1279	719

1 Calculation of RGB Values

The values of (x', y') obtained from Eq. 3 can be non-integral for particular values of (x, y). There are different ways in which the RGB pixel values for the non-integral values (x', y') can be obtained like rounding-off to nearest integer, interpolation etc. In this work we use the round-off method to get the RGB pixel values. In the round-off method, the values of (x', y') are rounded off to the nearest integer. For example for $(x', y') =$ (122.53, 176.21) we use $(x', y') \approx =$ (123, 176) for getting the pixel values. Since the round-off method produced good results, different techniques like interpolation were not used in this study.

2 Description of the Implementation

Here we describe the code for project the image in Fig. 1d on the PQRS frame in Fig. 1a. The same approach is used for projecting Fig. 1d on the PQRS frame in Figs. 1b and 1c.

i. In the first step, the images are loaded using *mpimg.imread*.

ii. In the next step, the method $getXXprimeFromPQRS$ is used to form the vectors C and C' from values of P, Q, R, S and P′, Q′, R′, S′.

iii. Then the homography matrix H in Eq. 1 is calculated in the method $calcHomography$ which uses as the input the vectors C and C'.

iv. Once the homography H is calculated, we now need to replace the pixel values within the PQRS polygon in Figs. 1a with the mapped values of pixel from Fig. 1d.

v. In order to do this, we need to identify the pixels in Fig. 1a which lie in PQRS polygon. This is done first by creating a blank (black color) copy of Fig. 1a using *np.zeros*. Then the pixel value for pixels which lie within PQRS are changed to white ([255,255,255]) using the $fillPoly$ method in *openCV*.

vi. We then loop over the pixels in Fig. 1a and for pixels $X_a = [x_a, y_a, 1]$ which lie in PQRS, we use the homography H and the equation $X_d = HX_a$ to get the corresponding pixel location $X_d = [x_d, y_d, 1]$ in Fig. 1d. The pixel value at location X_a in Fig. 1a is then replaced by the pixel value at location X_d in Fig. 1d.

The results for the projection of Fig. 1d on Figs. 1a, 1b and 1c is shown in Figs. 2, 3 and 4 respectively. We can see that Fig. 1d is projected very well on the PQRS frame.

(b) ***Find homographies between images shown in Figs. 1a and 1b and between images shown in Figs. 1b and 1c. Then apply the product of the two ho-***

Figure 2: Projection of Jackie's image on picture frame in Fig. 1a.

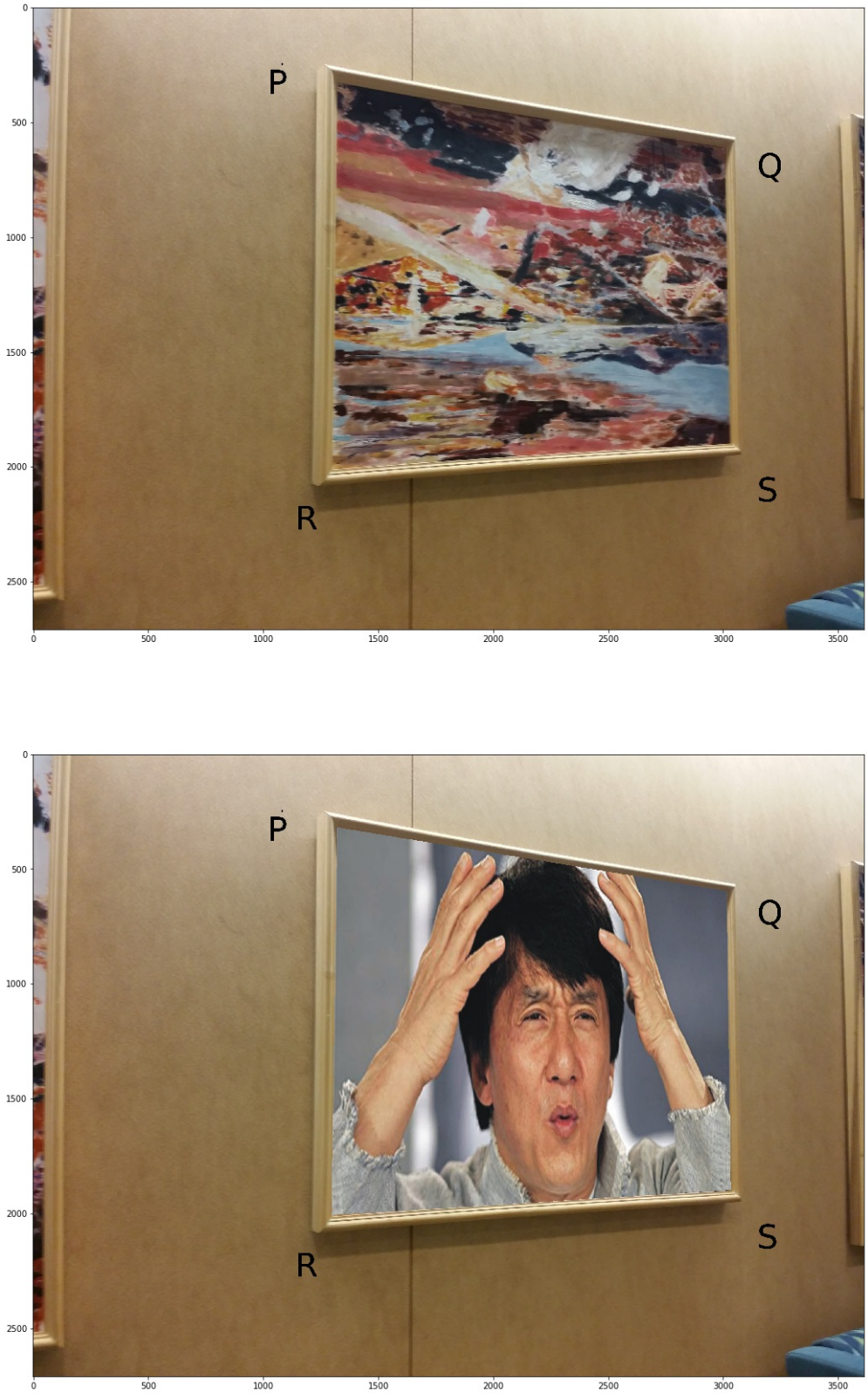

Figure 3: Projection of Jackie's image on picture frame in Fig. 1b.

Figure 4: Projection of Jackie's image on picture frame in Fig. 1c.

mographies to the image shown in Fig. 1a. The resulting image should look similar to the image shown in Fig. 1c

Solution:

3 Description of Implementation

i. First the homography H_{ab} between Fig. 1a and Fig. 1b is calculated using the *calcHomography* method described before so that $X_a = H_{ab}X_b$ for any pixel location $X_a = [x_a, y_a, 1]$ in Fig. 1a and $X_b = [x_b, y_b, 1]$ in Fig. 1b.

ii. Similarly the homography H_{bc} between Fig. 1b and Fig. 1c is calculated using the *calcHomography* method described before so that $X_c = H_{bc}X_b$ for any pixel location $X_b = [x_b, y_b, 1]$ in Fig. 1b and $X_c = [x_c, y_c, 1]$ in Fig. 1c.

iii. Next the homography $H_{ac} = H_{ab}H_{bc}$ using *numpy.matmul*.

iv. We then create a blank copy of Fig. 1c, loop over all the pixels in Fig. 1c and calculate the corresponding pixel location in Fig. 1a using $X_a = H_{ac}X_c$. The pixel value at X_c in the blank copy of Fig. 1c is then replaced by the pixel value at X_a. In this way Fig. 1a is projected on the plane in Fig. 1c.

The result of projection of image in Fig. 1a on the plane in Fig. 1c can be seen in Fig. 5.

(c) ***Repeat tasks 1(a) and 1(b) using your own images. You can capture three images of a planar surface from different viewpoints such as the ones in Figs. 1a, 1b and 1c. For the fourth image you can obtain a picture of a celebrity from the Internet or use a picture of your own face.***

Solution: The picture I have used for this part of the homework is Manchester United defender, Phil Jones as shown below in Fig. 6. The co-ordinates of corners of Jones's image P′, Q′, R′ and S′ and the corners of the frame P, Q, R, S shown in the top plot in Figs. 7, 8 and 9 is tabulated below.

Table 2: X and Y coordinates of points P, Q, R, S and P′, Q′, R′ S′

Fig. 7.

	x_i	y_i
P	1871	1053
Q	2490	1019
R	1881	2275
S	2463	2046

Fig. 8.

	x_i	y_i
P	1817	1103
Q	2699	1093
R	1851	2157
S	2685	2187

Fig. 9.

	x_i	y_i
P	1904	1140
Q	2524	1171
R	1921	2066
S	2510	2255

Fig. 6.

	x'_i	y'_i
P′	0	0
Q′	399	0
R′	0	399
S′	399	399

The three projections of Fig. 6 similar to Part 1(a) can be seen in Figs. 7, 8 and 9. The same code that has been used in Part 1(a) is used here. We can see that the projection is very good.

Fig. 10 shows the projection of the top image in Fig. 2 on the plane of the top image in Fig. 9. This is done using the code discussed in Part 1(b).

Figure 5: Projection of Fig. 1a on plane of Fig. 1c.

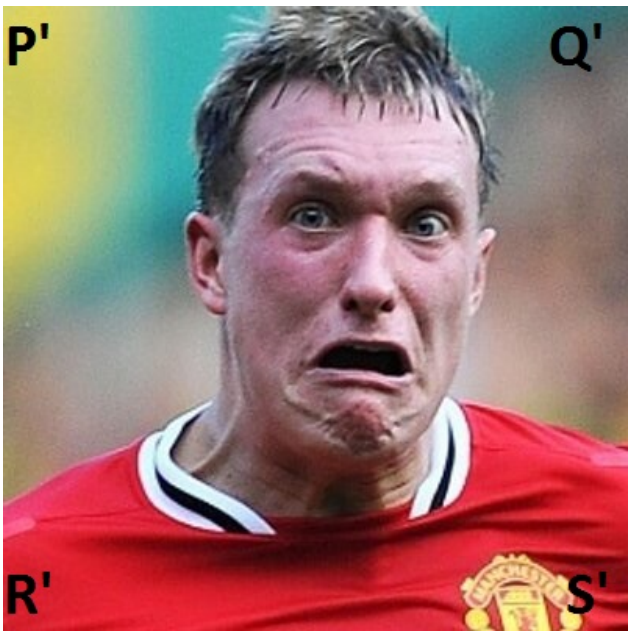

Figure 6: Phil Jones, Manchester United Defender.

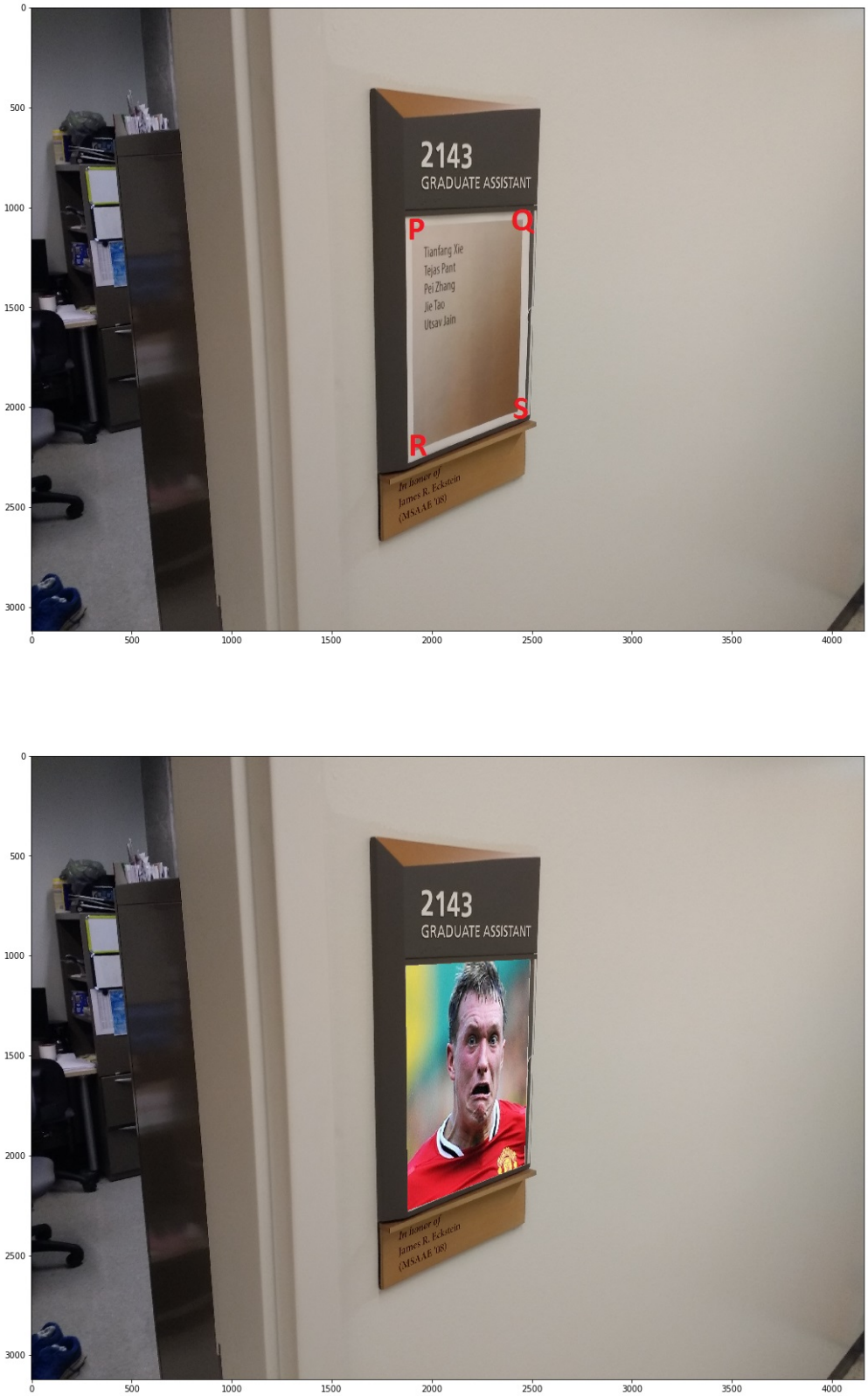

Figure 7: Projection of Phil Jones's image on planar surface similar to Fig. 1a.

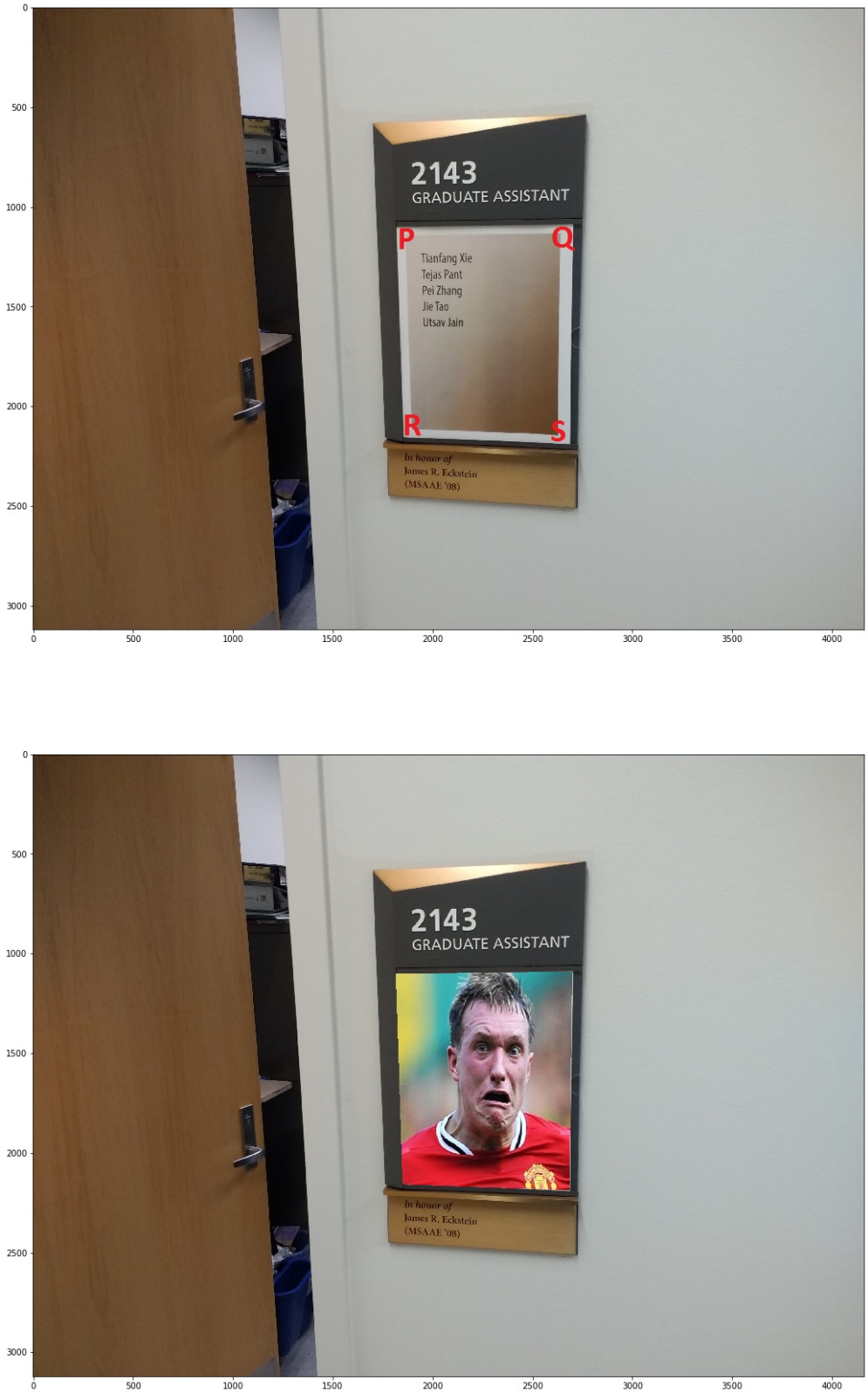

Figure 8: Projection of Phil Jones's image on planar surface similar to Fig. 1b.

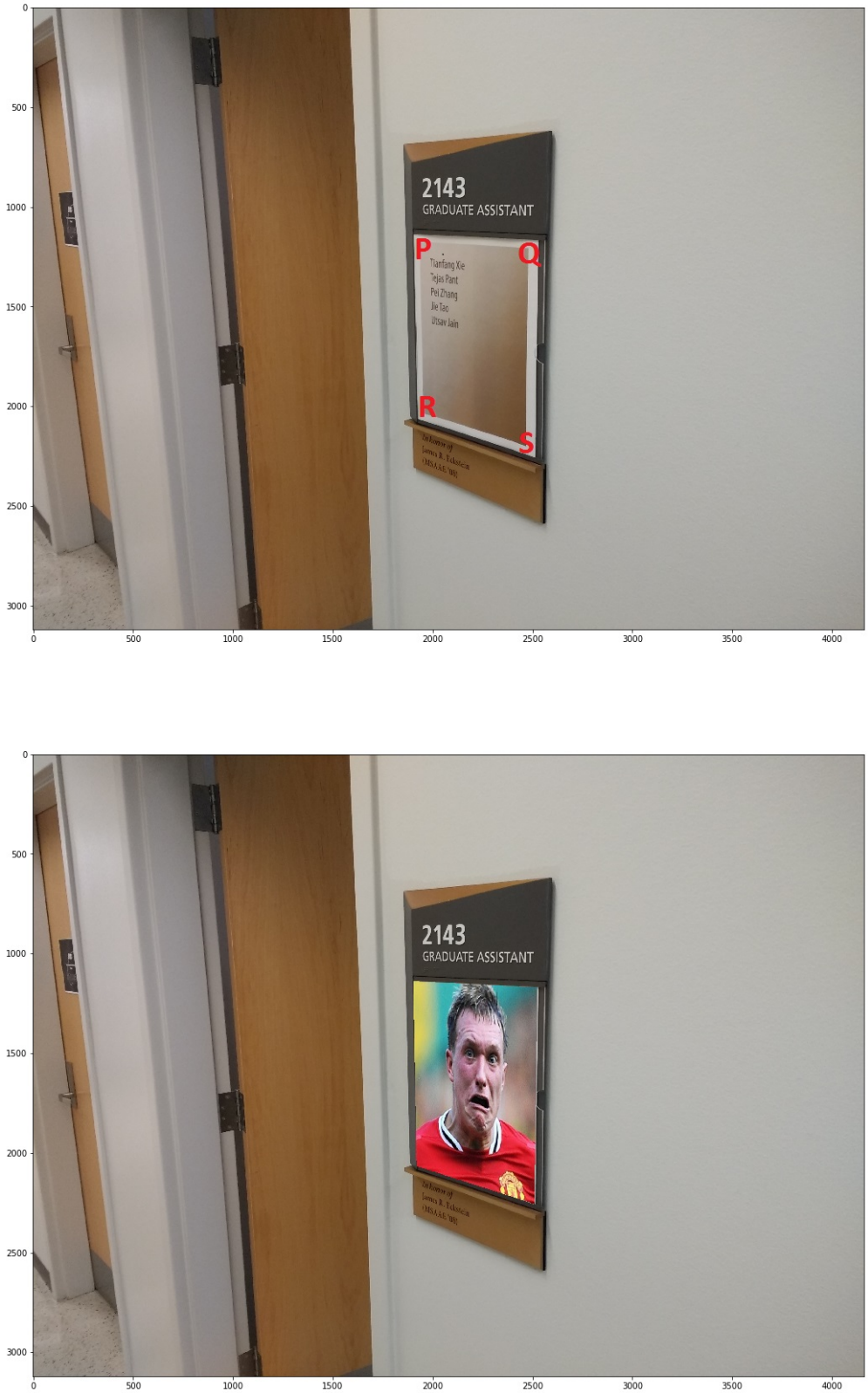

Figure 9: Projection of Phil Jones's image on planar surface similar to Fig. 1c.

Figure 10: Projection of top figure in Fig. 7 on plane of top figure in Fig. 9.

0.1 Problem 1(a)

In [1]:
```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import cv2
import glob
import os

os.listdir('../PicsHw2_org/')
images = glob.glob('../PicsHw2_org/*.jpg')

selectRangeImage = 3 #1 = 1a, 2 = 1b 3 = 1c
image = mpimg.imread(images[selectRangeImage-1])
image_org = np.copy(image)
image_jackie = mpimg.imread(images[3])

def getXXprimeFromPQRS(option):
    if option == 1:
        P = [1500, 112]
        Q = [2962, 730]
        R = [1485, 2243]
        S = [3015, 2054]
    elif option == 2:
        P = [1323, 321]
        Q = [3015, 598]
        R = [1299, 2007]
        S = [3033, 1898]
    elif option == 3:
        P = [916, 725]
        Q = [2809, 371]
        R = [890, 2080]
        S = [2847, 2231]

    #Jackie Image (Domain)
    P_ = [0,0]
    Q_ = [1279, 0]
    R_ = [0, 719]
    S_ = [1279, 719]

    #Vectors for caclulation of homography
    #Domain = Jackie, Range = 1abc
    #X_prime = P + Q + R + S #Range points
    #X = P_ + Q_ + R_ + S_ #Domain points

    #Domain = 1abc, Range = Jackie
    X = P + Q + R + S
    X_prime = P_ + Q_ + R_ + S_
```

```
    #Coordinates of frame corners in table format
    XYTablePQRS = np.array([P,R,S,Q],dtype='int32')
    return X_prime, X, XYTablePQRS

def calcHomography(X_prime,X):
#Caluclate the Homography H

    A = np.zeros((8,8))
    H = np.ones((3,3))

    for i in range(4):
        H_rowx = [0 for i in range(8)]
        H_rowy = [0 for i in range(8)]

        H_rowx[0] = X[2*i]
        H_rowx[1] = X[2*i+1]
        H_rowx[2] = 1
        H_rowx[6] = -X[2*i]*X_prime[2*i]
        H_rowx[7] = -X[2*i+1]*X_prime[2*i]

        H_rowy[3] = X[2*i]
        H_rowy[4] = X[2*i+1]
        H_rowy[5] = 1
        H_rowy[6] = -X[2*i]*X_prime[2*i+1]
        H_rowy[7] = -X[2*i+1]*X_prime[2*i+1]

        A[2*i] = H_rowx
        A[2*i+1] = H_rowy

    param = np.matmul(np.linalg.inv(A),np.array(X_prime))
    param = np.append(param,1)
    for i in range(3):
        H[i] = param[3*i:3*i+3]
    return H

X_prime_, X_, XYTablePQRS = getXXprimeFromPQRS(selectRangeImage)
H_ = calcHomography(X_prime_,X_)

image_empty = np.zeros(image.shape,dtype='uint8')
image_filled = cv2.fillPoly(image_empty,[XYTablePQRS],(255,255,255))

for i in range(0,image.shape[1]-1): #X-cordinate, col
    for j in range(0,image.shape[0]-1): #Y-coordinate, row
        if any(image_filled[j][i]) > 0:
            X_domain = [i,j]
            X_domain = np.append(np.array(X_domain),1)
            X_range = np.matmul(H_, X_domain)
```

```
                    X_range = np.rint(X_range/X_range[-1]) #Normalize and round off to nearest i
                    X_range = X_range.astype(int)
                    if(X_range[0] < image_jackie.shape[1] and \
                       X_range[1] < image_jackie.shape[0]):
                        image[j][i] = image_jackie[X_range[1]][X_range[0]]
```

```
In [ ]: plt.figure(figsize=(30,30))
        plt.subplot(211)
        plt.imshow(image_org)
        plt.subplot(212)
        plt.imshow(image)
```

0.2 Problem 1(b)

```
In [1]: import matplotlib.image as mpimg
        import matplotlib.pyplot as plt
        import numpy as np
        import cv2
        import glob
        import os

        os.listdir('../PicsHw2_org/')
        images = glob.glob('../PicsHw2_org/*.jpg')
        image1a = mpimg.imread(images[0])
        image1c = mpimg.imread(images[2])

        def calcHomography(X_prime,X):
        #Caluclate the Homography H

            A = np.zeros((8,8))
            H = np.ones((3,3))

            for i in range(4):
                H_rowx = [0 for i in range(8)]
                H_rowy = [0 for i in range(8)]

                H_rowx[0] = X[2*i]
                H_rowx[1] = X[2*i+1]
                H_rowx[2] = 1
                H_rowx[6] = -X[2*i]*X_prime[2*i]
                H_rowx[7] = -X[2*i+1]*X_prime[2*i]

                H_rowy[3] = X[2*i]
                H_rowy[4] = X[2*i+1]
                H_rowy[5] = 1
                H_rowy[6] = -X[2*i]*X_prime[2*i+1]
                H_rowy[7] = -X[2*i+1]*X_prime[2*i+1]
```

```
            A[2*i] = H_rowx
            A[2*i+1] = H_rowy

        param = np.matmul(np.linalg.inv(A),np.array(X_prime))
        param = np.append(param,1)
        for i in range(3):
            H[i] = param[3*i:3*i+3]
        return H
```

In [2]:
```
#Homography between 1a and 1b
Pa = [1500, 112]
Qa = [2962, 730]
Ra = [1485, 2243]
Sa = [3015, 2054]

Pb = [1323, 321]
Qb = [3015, 598]
Rb = [1299, 2007]
Sb = [3033, 1898]

Pc = [916, 725]
Qc = [2809, 371]
Rc = [890, 2080]
Sc = [2847, 2231]

#Homography between 1a and 1b
X_prime_ = Pa + Qa + Ra + Sa
X_ = Pb + Qb + Rb + Sb
Hab = calcHomography(X_prime_,X_)

#Homography between 1b and 1c
X_prime_ = Pb + Qb + Rb + Sb
X_ = Pc + Qc + Rc + Sc
Hbc = calcHomography(X_prime_,X_)
Hac = np.matmul(Hab,Hbc)
```

In [ ]:
```
image_1c_generated = np.zeros(image1c.shape,dtype='uint8')
for i in range(0,image1c.shape[1]-1): #X-cordinate, col
    for j in range(0,image1c.shape[0]-1): #Y-coordinate, row
        X_domain = [i,j]
        X_domain = np.append(np.array(X_domain),1)
        X_range = np.matmul(Hac, X_domain)
        X_range = np.rint(X_range/X_range[-1]) #Normalize and round off to nearest integ
        X_range = X_range.astype(int)
        if(X_range[0] > 0 and X_range[1] > 0 and \
           X_range[0] < image1a.shape[1] and \
           X_range[1] < image1a.shape[0]):
            image_1c_generated[j][i] = image1a[X_range[1]][X_range[0]]
```

```
In [ ]: plt.figure(figsize=(20,20))
        plt.subplot(121)
        plt.title('Fig. 1(a)',fontsize=18)
        plt.imshow(image1a)
        plt.subplot(122)
        plt.title('Fig. 1(c)',fontsize=18)
        plt.imshow(image1c)
        plt.figure(figsize=(15,15))
        plt.title('Projection of Fig. 1(a) on plane of Fig. 1(c)',fontsize=18)
        plt.imshow(image_1c_generated)
```

0.3 Problem 1c

```
In [1]: import matplotlib.image as mpimg
        import matplotlib.pyplot as plt
        import numpy as np
        import cv2
        import glob
        import os

        images = glob.glob('../Pics_mine/*.jpg')

        selectRangeImage = 3 #1 = 1a, 2 = 1b 3 = 1c
        image = mpimg.imread(images[selectRangeImage])
        image_org = np.copy(image)
        image_jones = mpimg.imread(images[0])

        def getXXprimeFromPQRS(option):
            if option == 1:
                P = [1871, 1053]
                Q = [2490, 1019]
                R = [1881, 2275]
                S = [2463, 2046]
            elif option == 2:
                P = [1817, 1103]
                Q = [2699, 1093]
                R = [1851, 2157]
                S = [2685, 2187]
            elif option == 3:
                P = [1904, 1140]
                Q = [2524, 1171]
                R = [1921, 2066]
                S = [2510, 2255]

            #Jones Image (Domain)
            P_ = [0,0]
            Q_ = [399, 0]
            R_ = [0, 399]
```

```
    S_ = [399, 399]

    #Domain = 1abc, Range = Jones
    X = P + Q + R + S
    X_prime = P_ + Q_ + R_ + S_

    #Coordinates of frame corners in table format
    XYTablePQRS = np.array([P,R,S,Q],dtype='int32')
    return X_prime, X, XYTablePQRS

def calcHomography(X_prime,X):
#Caluclate the Homography H

    A = np.zeros((8,8))
    H = np.ones((3,3))

    for i in range(4):
        H_rowx = [0 for i in range(8)]
        H_rowy = [0 for i in range(8)]

        H_rowx[0] = X[2*i]
        H_rowx[1] = X[2*i+1]
        H_rowx[2] = 1
        H_rowx[6] = -X[2*i]*X_prime[2*i]
        H_rowx[7] = -X[2*i+1]*X_prime[2*i]

        H_rowy[3] = X[2*i]
        H_rowy[4] = X[2*i+1]
        H_rowy[5] = 1
        H_rowy[6] = -X[2*i]*X_prime[2*i+1]
        H_rowy[7] = -X[2*i+1]*X_prime[2*i+1]

        A[2*i] = H_rowx
        A[2*i+1] = H_rowy

    param = np.matmul(np.linalg.inv(A),np.array(X_prime))
    param = np.append(param,1)
    for i in range(3):
        H[i] = param[3*i:3*i+3]
    return H

X_prime_, X_, XYTablePQRS = getXXprimeFromPQRS(selectRangeImage)
H_ = calcHomography(X_prime_,X_)

image_empty = np.zeros(image.shape,dtype='uint8')
image_filled = cv2.fillPoly(image_empty,[XYTablePQRS],(255,255,255))

for i in range(0,image.shape[1]-1): #X-cordinate, col
```

```
        for j in range(0,image.shape[0]-1): #Y-coordinate, row
            if any(image_filled[j][i]) > 0:
                X_domain = [i,j]
                X_domain = np.append(np.array(X_domain),1)
                X_range = np.matmul(H_, X_domain)
                X_range = np.rint(X_range/X_range[-1]) #Normalize and round off to nearest i
                X_range = X_range.astype(int)
                if(X_range[0] < image_jones.shape[1] and \
                   X_range[1] < image_jones.shape[0]):
                    image[j][i] = image_jones[X_range[1]][X_range[0]]
```

```
In [ ]: plt.figure(figsize=(30,30))
        plt.subplot(211)
        plt.imshow(image_org)
        plt.subplot(212)
        plt.imshow(image)
```

0.4 Problem 1c , Solution to 1b Task

```
In [1]: import matplotlib.image as mpimg
        import matplotlib.pyplot as plt
        import numpy as np
        import cv2
        import glob
        import os

        images = glob.glob('../Pics_mine/*.jpg')
        image1a = mpimg.imread(images[1])
        image1c = mpimg.imread(images[3])

        def calcHomography(X_prime,X):
        #Caluclate the Homography H

            A = np.zeros((8,8))
            H = np.ones((3,3))

            for i in range(4):
                H_rowx = [0 for i in range(8)]
                H_rowy = [0 for i in range(8)]

                H_rowx[0] = X[2*i]
                H_rowx[1] = X[2*i+1]
                H_rowx[2] = 1
                H_rowx[6] = -X[2*i]*X_prime[2*i]
                H_rowx[7] = -X[2*i+1]*X_prime[2*i]

                H_rowy[3] = X[2*i]
```

```
            H_rowy[4] = X[2*i+1]
            H_rowy[5] = 1
            H_rowy[6] = -X[2*i]*X_prime[2*i+1]
            H_rowy[7] = -X[2*i+1]*X_prime[2*i+1]

            A[2*i] = H_rowx
            A[2*i+1] = H_rowy

        param = np.matmul(np.linalg.inv(A),np.array(X_prime))
        param = np.append(param,1)
        for i in range(3):
            H[i] = param[3*i:3*i+3]
        return H
```

In [2]:
```
#Homography between 1a and 1b
Pa = [1871, 1053]
Qa = [2490, 1019]
Ra = [1881, 2275]
Sa = [2463, 2046]

Pb = [1817, 1103]
Qb = [2699, 1093]
Rb = [1851, 2157]
Sb = [2685, 2187]

Pc = [1904, 1140]
Qc = [2524, 1171]
Rc = [1921, 2066]
Sc = [2510, 2255]

#Homography between 1a and 1b
X_prime_ = Pa + Qa + Ra + Sa
X_ = Pb + Qb + Rb + Sb
Hab = calcHomography(X_prime_,X_)

#Homography between 1b and 1c
X_prime_ = Pb + Qb + Rb + Sb
X_ = Pc + Qc + Rc + Sc
Hbc = calcHomography(X_prime_,X_)
Hac = np.matmul(Hab,Hbc)
```

In [3]:
```
image_1c_generated = np.zeros(image1c.shape,dtype='uint8')
for i in range(0,image1c.shape[1]-1): #X-cordinate, col
    for j in range(0,image1c.shape[0]-1): #Y-coordinate, row
        X_domain = [i,j]
        X_domain = np.append(np.array(X_domain),1)
        X_range = np.matmul(Hac, X_domain)
        X_range = np.rint(X_range/X_range[-1]) #Normalize and round off to nearest integ
```

```
            X_range = X_range.astype(int)
            if(X_range[0] > 0 and X_range[1] > 0 and \
               X_range[0] < image1a.shape[1] and \
               X_range[1] < image1a.shape[0]):
                image_1c_generated[j][i] = image1a[X_range[1]][X_range[0]]
```

In [ ]:
```
plt.figure(figsize=(20,20))
plt.subplot(121)
plt.title('My Image Similar to Fig. 1(a)',fontsize=18)
plt.imshow(image1a)
plt.subplot(122)
plt.title('My Image Similar to Fig. 1(c)',fontsize=18)
plt.imshow(image1c)
plt.figure(figsize=(15,15))
plt.title('Projection',fontsize=18)
plt.imshow(image_1c_generated)
```