ECE661HW2

Chang Yang

September 3, 2018

The code is pasted at the end of the report

1 Task 1

1.1 Task 1(a)

1.1.1 find pixel coordinates

To do this task, we first need to find the pixel of PQRS in Fig. 1(a). Importing the figure into GIMP, then click at point P, the coordinate of P will show on the left down side of the GIMP.

As shown above, the position of the point P is (1513, 182). Similarly, Q is (2948, 726), R is (1496, 2244), S is (2998, 2046).

As for Fig. 1(d), we first draw a bounding box surrounding Jackie Chan's face, like below.

GIMP will give the Position and Size information for this bounding box, which is (354, 180), (575,525) repectively. Then we know that P' is (354, 180), Q' is (929, 180), R' is (354, 705), S' is (929, 705) using above information.

1.1.2 Calculating Homography

Then we need to calculate the homography between Fig. 1(d) and Fig. 1(a).

To find the Homography $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$. We can set $h_{33} = 1$. Then notice projecting a physical point $a = (x, y, 1)^T$ to another physical point $b = (x', y', 1)^H$ using Homography above can be related as below:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Solving above equation, we get

$$x' = h_{11}x + h_{12}y + h_{13} \tag{1}$$

$$y' = h_{21}x + h_{22}y + h_{23} \tag{2}$$

$$1 = h_{31}x + h_{32}y + 1 \tag{3}$$

Divide Eq. (1) and Eq. (2) by Eq. (3), respectively, we get,

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1} \tag{4}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1} \tag{5}$$

Rewrite above equations as

$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' = x' \tag{6}$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' = y' \tag{7}$$

Above can be written as matrix equation form as

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -yx' \\ 0 & 0 & 0 & x & y & 1 & -xy' & -yy' \end{bmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

Using four groups of points, above equation will be

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{pmatrix}$$

record above equation as $Ah = b$. Thus, $h = A^{-1}b$

Thus, the matrix equation of transform (P',Q',R',S') to (P,Q,R,S) is

$$\begin{bmatrix} x'_P & y'_P & 1 & 0 & 0 & 0 & -x'_Px_P & -y'_Px_P \\ 0 & 0 & 0 & x'_P & y'_P & 1 & -x'_Py_P & -y'_Py_P \\ x'_Q & y'_Q & 1 & 0 & 0 & 0 & -x'_Qx_Q & -y'_Qx_Q \\ 0 & 0 & 0 & x'_Q & y'_Q & 1 & -x'_Qy_Q & -y'_Qy_Q \\ x'_R & y'_R & 1 & 0 & 0 & 0 & -x'_Rx_R & -y'_Rx_R \\ 0 & 0 & 0 & x'_R & y'_R & 1 & -x'_Ry_R & -y'_Ry_R \\ x'_S & y'_S & 1 & 0 & 0 & 0 & -x'_Sx_S & -y'_Sx_S \\ 0 & 0 & 0 & x'_S & y'_S & 1 & -x'_Sy_S & -y'_Sy_S \end{bmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x_P \\ y_P \\ x_Q \\ y_Q \\ x_R \\ y_R \\ x_S \\ y_S \end{pmatrix}$$

1.1.3 apply Homography to project the image

Using the python code attached at the end of report. We can finally get the Fig.1(a) as

Do the same thing for Fig.1(b), we can get

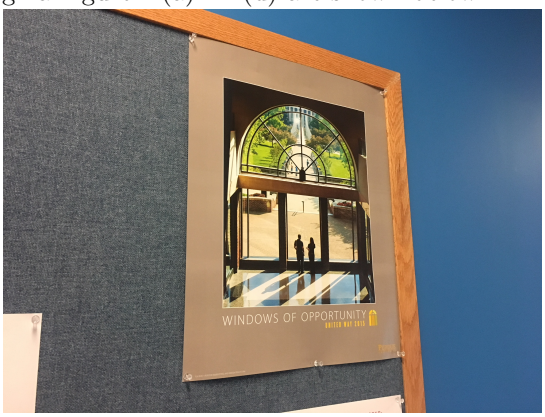
And for Fig.1(c), we can get

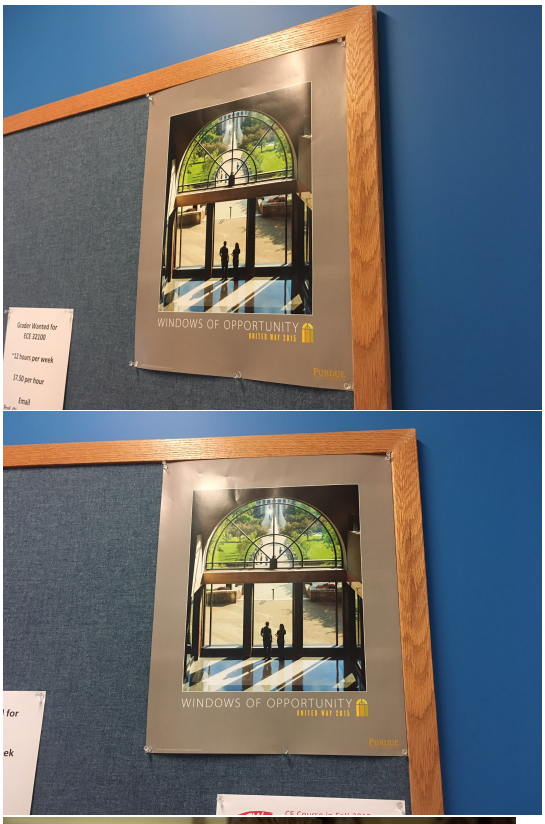
1.2 Task 1(b)

Apply the product of two homographies to Fig. 1(a), we can get below picture.

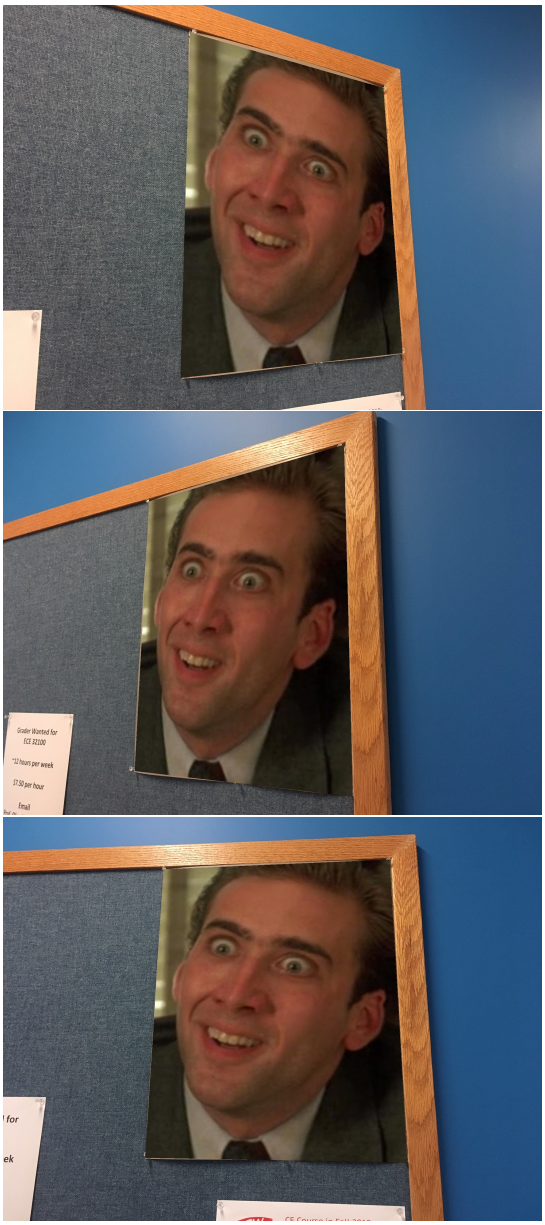
2 Task 2

Original figure 2.(a) - 2.(d) are shown below:





After transforming, the pictures look like:



2.1 Task 2(b)

Apply the product of two homographies to Fig. 2(a), we can get below picture.

3 The code

3.1 Task1.py

```
import numpy as np
import cv2
from calHomography import *
from projectimage import *

#Task 1(a)
#NOTICE if using GIMP to get coordinates of points, the (x, y) is different from
P = Point(182, 1513)
Q = Point(726, 2948)
R = Point(2244, 1496)
S = Point(2046, 2998)
quad = Quad(P, Q, R, S)

P2 = Point(347, 1326)
Q2 = Point(627, 3014)
R2 = Point(2019, 1298)
S2 = Point(1898, 3025)
quad2 = Quad(P2, Q2, R2, S2)

P3 = Point(743, 913)
Q3 = Point(374, 2811)
R3 = Point(2079, 902)
S3 = Point(2228, 2838)
quad3 = Quad(P3, Q3, R3, S3)

Pp = Point(180, 354)
Qp = Point(180, 929)
```

```
Rp = Point(705, 354)
Sp = Point(705, 929)
quadp = Quad(Pp, Qp, Rp, Sp)

#read image
img1 = cv2.imread('./PicsHw2/1.jpg')
img2 = cv2.imread('./PicsHw2/2.jpg')
img3 = cv2.imread('./PicsHw2/3.jpg')
imgj = cv2.imread('./PicsHw2/Jackie.jpg')
DoProject(img1, imgj, quad, quadp)
DoProject(img2, imgj, quad2, quadp)
DoProject(img3, imgj, quad3, quadp)
cv2.imwrite('1j.jpg',img1)
cv2.imwrite('2j.jpg',img2)
cv2.imwrite('3j.jpg',img3)
cv2.destroyAllWindows()


#Task 1(b)
Hab = cal_Homography(quad2, quad)
Hbc = cal_Homography(quad3, quad2)
H = np.matmul(Hab, Hbc)
img1 = cv2.imread('./PicsHw2/1.jpg')
img3p = np.zeros(img1.shape, np.uint8)
#to project the image
x0, x1, y0, y1 = quad.get_box()

for x in range(x0,x1+1):
        for y in range(y0,y1+1):
                pt = Point(x,y)
                if is_inside(quad, pt):
                        ptp = np.matmul(H, pt.hp.reshape((3,1)))
                        ptp = ptp / ptp[2][0]
                        #ptp = Point(ptp[0][0], ptp[1][0])
                        ptpx = int(round(ptp[0][0]))
                        ptpy = int(round(ptp[1][0]))
                        img3p[ptpx, ptpy] = getRGB(pt, img1)

cv2.imwrite('3p.jpg',img3p)
cv2.destroyAllWindows()
```

3.2 Task2.py

```
import numpy as np
import cv2
from calHomography import *
```

```
from projectimage import *

#Task 2(a)
#NOTICE if using GIMP to get coordinates of points, the (x, y) is different from
P = Point(102, 699)
Q = Point(309, 1422)
R = Point(1392, 663)
S = Point(1299, 1494)
quad = Quad(P, Q, R, S)

P2 = Point(339, 546)
Q2 = Point(126, 1275)
R2 = Point(1347, 492)
S2 = Point(1443, 1308)
quad2 = Quad(P2, Q2, R2, S2)

P3 = Point(195, 600)
Q3 = Point(159, 1446)
R3 = Point(1275, 531)
S3 = Point(1293, 1512)
quad3 = Quad(P3, Q3, R3, S3)

Pp = Point(12, 692)
Qp = Point(12, 1262)
Rp = Point(790, 692)
Sp = Point(790, 1262)
quadp = Quad(Pp, Qp, Rp, Sp)

#read image
img1 = cv2.imread('./PicsHw2/4.jpg')
img2 = cv2.imread('./PicsHw2/5.jpg')
img3 = cv2.imread('./PicsHw2/6.jpg')
imgj = cv2.imread('./PicsHw2/NicolasCage.jpg')
DoProject(img1, imgj, quad, quadp)
DoProject(img2, imgj, quad2, quadp)
DoProject(img3, imgj, quad3, quadp)
cv2.imwrite('4j.jpg',img1)
cv2.imwrite('5j.jpg',img2)
cv2.imwrite('6j.jpg',img3)
cv2.destroyAllWindows()


#Task 1(b)
Hab = cal_Homography(quad2, quad)
Hbc = cal_Homography(quad3, quad2)
H = np.matmul(Hab, Hbc)
```

```
img1 = cv2.imread('./PicsHw2/4.jpg')
img3p = np.zeros(img1.shape, np.uint8)
#to project the image
x0, x1, y0, y1 = quad.get_box()

for x in range(x0,x1+1):
        for y in range(y0,y1+1):
                pt = Point(x,y)
                if is_inside(quad, pt):
                        ptp = np.matmul(H, pt.hp.reshape((3,1)))
                        ptp = ptp / ptp[2][0]
                        #ptp = Point(ptp[0][0], ptp[1][0])
                        ptpx = int(round(ptp[0][0]))
                        ptpy = int(round(ptp[1][0]))
                        img3p[ptpx, ptpy] = getRGB(pt, img1)

cv2.imwrite('6p.jpg',img3p)
cv2.destroyAllWindows()
```

3.3 projectimage.py

```
import numpy as np
import cv2
from calHomography import *

#decide is p is in PQRS of image 1
def is_inside(quad, p):
        y0 = (- quad.edges[0][0] * p.x - quad.edges[0][2]) / quad.edges[0][1]
        y1 = (- quad.edges[2][0] * p.x - quad.edges[2][2]) / quad.edges[2][1]
        x1 = (- quad.edges[1][1] * p.y - quad.edges[1][2]) / quad.edges[1][0]
        x0 = (- quad.edges[3][1] * p.y - quad.edges[3][2]) / quad.edges[3][0]

        if p.x >= x0 and p.x <= x1 and p.y >= y0 and p.y <= y1 :
                return True
        return False

#get RGB when p is not a integer pixel point
def getRGB(pt, img):
        x = int(round(pt.x))
        y = int(round(pt.y))
        return img[x, y]

def DoProject(img, imgp, quad, quadp):
        #First Calculate the Homography from quad to quadq
        H = cal_Homography(quadp, quad)
```

```
        #to project the image
        x0, x1, y0, y1 = quad.get_box()

        for x in range(x0,x1+1):
                for y in range(y0,y1+1):
                        pt = Point(x,y)
                        if is_inside(quad, pt):
                                ptp = np.matmul(H, pt.hp.reshape((3,1)))
                                ptp = ptp / ptp[2][0]
                                ptp = Point(ptp[0][0], ptp[1][0])
                                img[x,y] = getRGB(ptp, imgp)
```

3.4 calHomography.py

```
import numpy as np
from numpy.linalg import inv

class Point():
        """docstring for point"""
        def __init__(self, x, y):
                self.x = x
                self.y = y
                self.hp = np.array([x,y,1])

class Quad():
        """docstring for rectangle"""
        def __init__(self, P, Q, R, S):
                self.A = [P, Q, R, S]
                self.edges = [np.cross(P.hp, R.hp), np.cross(R.hp, S.hp), np.cro

        def get_box(self):
                x0 = self.A[0].x
                x1 = self.A[0].x
                y0 = self.A[0].y
                y1 = self.A[0].y
                for i in range(4):
                        if self.A[i].x < x0:
                                x0 = self.A[i].x
                        if self.A[i].x > x1:
                                x1 = self.A[i].x
                        if self.A[i].y < y0:
                                y0 = self.A[i].y
                        if self.A[i].y > y1:
                                y1 = self.A[i].y
                return x0, x1, y0, y1
```

```
def cal_Homo_2l(p0, p1):
        A = np.zeros((2,8))

        A[0][0] = p1.x
        A[0][1] = p1.y
        A[0][2] = 1
        A[0][6] = - p1.x * p0.x
        A[0][7] = - p1.y * p0.x

        A[1][3] = p1.x
        A[1][4] = p1.y
        A[1][5] = 1
        A[1][6] = - p1.x * p0.y
        A[1][7] = - p1.y * p0.y

        b = np.zeros((2,1))
        b[0] = p0.x
        b[1] = p0.y

        return A, b


def cal_Homography(quad0, quad1):
        #this function computes the projection H from (Pc, Qc, Rc, Sc) to (P, Q,
        #create left hand side
        A, b = cal_Homo_2l(quad0.A[0], quad1.A[0])
        for i in range(3):
                C, d = cal_Homo_2l(quad0.A[i+1], quad1.A[i+1])
                A = np.concatenate((A,C), axis = 0)
                b = np.concatenate((b,d), axis = 0)

        Ainv = inv(A)
        h = np.matmul(Ainv, b)
        h = np.append(h,[[1]])
        h = np.reshape(h,(3,3))
        return h
```