**ECE 661 Homework 3**

**Weichen Xu**

**xu1363@purdue.edu**

**1. Methods and Implementations**

**Point-to-point correspondence**

We usually obtain the homography matrix by using point pairs in both images.

Since we know that a typical representation of homography matrix is a 3\*3 matrix with the to be 1.

Applying the homography matrix to the mapping formula, we can get following result

If we set and , then we can get following equations

Further we can further transform to following equations

In total, the H matrix has 8 unknown elements, so we need at least 4 sets of corresponding point pairs and with 8 equations to solve the equations.

We can then obtain h by multiplying the inverse matrix of A with b

**Two-step method**

**Step 1: Remove projective distortion**

The first step of two-step method is removing projective distortion using the vanishing line method. By finding the vanishing line, we can get the homography mapping vanishing line in image domain to the in world domain.

First, we obtain two pairs of parallel lines in the image by setting four pairs of points at the parallel lines.

Then we can obtain two point at infinity (ideal point) by two pairs of parallel lines.

Once we get two ideal points, we can obtain the vanishing line.

Finally, the homography removing the projective distortion will be denoted as

**Step 2: Remove affine distortion**

To remove the affine distortion, we need at least two pairs orthogonal lines in the physical world to remove the affine distortion. Let denote we have two parallel lines and , then we can get the angle between two lines.

We can define a homography matrix that mapping the world plane to the image plane, denoted as . Then we can get equations of and . When the two lines are orthogonal, we assign , we get following equation.

While

Transform the equation to the following form.

Let denote , we can then obtain

Since only the ratio matters, we can assign . We can use singular value decomposition (SVD) to get A using following relations.

Finally, by taking the inverse of , we can get the homography mapping from the image plane to the world plane.

**One-step method**

We can write a homography matrix in the general form

The dual conic in the image plane is given by

If we have two orthogonal lines in the world plane, and , we will have following.

Since only the ratio matters, we set , then we still have 5 unknowns remaining. So, we need at least 5 pairs of equations to solve all the unknowns.

After we solve the equations, we can apply SVD to get , then we can also get . We can then get the homography matrix to remove the projective and affine distortion in one step.

**2. Input and result images**

**Task 1**

A picture containing sitting, person, large, light

Description automatically generated

Figure 1. Img1 original image, yellow points are used for point-to-point correspondence, red points are used for two-step method, red point and blue points together are used for one-step method

A car parked in front of a building

Description automatically generated

Figure 2. Img1 after point-to-point correspondence

The roof of a building

Description automatically generatedFigure 3. Img1 after projective correction of two-step method

A close up of a building

Description automatically generated Figure 4. Img1 after affine correction of two-step method

A picture containing field

Description automatically generated

Figure 5. Img2 original image, yellow points are used for point-to-point correspondence, red points are used for two-step method, red point and blue points together are used for one-step method

A close up of a logo

Description automatically generated

Figure 6. Img2 after point-to-point correspondence

A picture containing road, person

Description automatically generated

Figure 7. Img2 after projective correction of two-step method



Figure 8. Img2 after affine correction of two-step method



Figure 9. Img2 after one-step method

A close up of a computer

Description automatically generated

Figure 10. Img3 original image, red points are used for point-to-point correspondence and two-step method, red point and blue points together are used for one-step method

A close up of a computer

Description automatically generated

Figure 11. Img3 after point-to-point correspondence

A picture containing indoor, bed, computer, room

Description automatically generated

Figure 12. Img3 after projective correction of two-step method

**Task 2**

A picture containing indoor, photo, sitting, refrigerator

Description automatically generated

Figure 13. myimg1 original image, red points are used for point-to-point correspondence and two-step method, red point and blue points together are used for one-step method

A picture containing indoor, dark, building, window

Description automatically generated

Figure 14. myimg1 after point-to-point correspondence

A close up of a piano

Description automatically generated

Figure 15. myimg1 after projective correction of two-step method

A close up of a window

Description automatically generated

Figure 16. myimg1 after affine correction of two-step method

A picture containing circuit

Description automatically generated

Figure 17. myimg1 original image, red points are used for point-to-point correspondence and two-step method, red point and blue points together are used for one-step method

A close up of a device

Description automatically generated

Figure 18. myimg2 after point-to-point correspondence

A picture containing circuit

Description automatically generated

Figure 19. myimg2 after projective correction of two-step method

A picture containing circuit

Description automatically generated

Figure 20. myimg2 after affine correction of two-step method

A close up of a device

Description automatically generated

Figure 21. myimg2 after one-step method

**3. Observation**

In my experiments, I think point-to-point correspondence method is very straightforward and works well, the only thing that makes it not practical is that you need to know the image’s coordinates in real world plane.

The two-step method in this case are most complex method in all three methods since you need to have obtained two homography matrix. However, comparing to the one-step method, the two-step method tends to have more robust performance regarding the choice of points.

In my experiments, only one image out of five images has reasonable result using one-step method, others basically fail to show any pattern or even get dramatic in the output image size. The one-step method is very sensitive to points, even though very slight change can dramatically change the homography matrix, which makes it hard to implement, because it really needs much trial and error for a set of good points.

**4. Code**

#!/usr/bin/env python

# coding: utf-8

import cv2

import matplotlib.pyplot as plt

import numpy as np

def get\_homography(target,source):

A = np.zeros((8,8))

b = np.reshape(target,(8,1))

for i in range(4):

# define corresponding points pair of PQSR

x1 = source[i,0]

y1 = source[i,1]

x2 = target[i,0]

y2 = target[i,1]

# build the row of matrix A

A[i\*2,:] = [x1,y1,1,0,0,0,-x1\*x2,-y1\*x2]

A[i\*2+1,:] = [0,0,0,x1,y1,1,-x1\*y2,-y1\*y2]

# build the element of vector b

b[i\*2] = x2

b[i\*2+1] = y2

# multiply the inverse matrix of A with b to get the vector of all coefficients of H

h = np.matmul(np.linalg.inv(A),b)

# attach the last scalar of H, which is 1

h = np.append(h,1)

# resize to 3\*3 matrix

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

return H

def mapping(img\_target,H):

P\_distort = np.array([0,0,1])

Q\_distort = np.array([0,img\_target.shape[0]-1,1])

R\_distort = np.array([img\_target.shape[1]-1,img\_target.shape[0]-1,1])

S\_distort = np.array([img\_target.shape[1]-1,0,1])

P\_world = np.matmul(H,P\_distort)

P\_world = P\_world / P\_world[2]

Q\_world = np.matmul(H,Q\_distort)

Q\_world = Q\_world / Q\_world[2]

R\_world = np.matmul(H,R\_distort)

R\_world = R\_world / R\_world[2]

S\_world = np.matmul(H,S\_distort)

S\_world = S\_world / S\_world[2]

xmin = np.int32(np.round(np.amin([P\_world[0],Q\_world[0],R\_world[0],S\_world[0]])))

xmax = np.int32(np.ceil(np.amax([P\_world[0],Q\_world[0],R\_world[0],S\_world[0]])))

ymin = np.int32(np.round(np.amin([P\_world[1],Q\_world[1],R\_world[1],S\_world[1]])))

ymax = np.int32(np.ceil(np.amax([P\_world[1],Q\_world[1],R\_world[1],S\_world[1]])))

xlen = xmax-xmin

ylen = ymax-ymin

img\_new = np.zeros((ylen,xlen,3), dtype=np.uint8)

#print(xmin,ymin)

#print('The output image size is',xlen,ylen)

Hinv = np.linalg.inv(H)

# get the corresponding location after applying homography

for i in range(xlen):

for j in range(ylen):

input = np.array([i+xmin,j+ymin,1])

output = np.matmul(Hinv,input)

x = np.int(np.round(output[0]/output[2]))

y = np.int(np.round(output[1]/output[2]))

if x>0 and x<img\_target.shape[1]-1 and y>0 and y<img\_target.shape[0]-1:

# replace the pixel value with corresponding points from source image

img\_new[j,i,:] = img\_target[y,x,:]

return img\_new

def get\_projective(corners):

# calculate parallel lines

l11 = np.cross(corners[0], corners[1])

l12 = np.cross(corners[2], corners[3])

# calculate parallel ideal point

pt1 = np.cross(l11, l12)

pt1 = pt1 / pt1[2]

l21 = np.cross(corners[0], corners[3])

l22 = np.cross(corners[1], corners[2])

pt2 = np.cross(l21, l22)

pt2 = pt2 / pt2[2]

# get vanishing line from ideal points

VL = np.cross(pt1, pt2)

VL = VL / VL[2]

H = np.array([[1, 0, 0], [0, 1, 0], [VL[0], VL[1], VL[2]]])

return H

def get\_affine(corners):

# calculate parallel lines

l1 = np.cross(corners[0],corners[1])

l1 = l1 / l1[2]

l2 = np.cross(corners[1],corners[2])

l2 = l2 / l2[2]

m1 = np.cross(corners[0],corners[3])

m1 = m1 / m1[2]

m2 = np.cross(corners[2],corners[3])

m2 = m2 / m2[2]

# solving unknowns in S from parallel lines

A = np.array([[l1[0] \* m1[0], l1[0] \* m1[1] + l1[1] \* m1[0]], [l2[0] \* m2[0], l2[0] \* m2[1] + l2[1] \* m2[0]]])

b = np.array([[-l1[1] \* m1[1]], [-l2[1] \* m2[1]]])

S = np.zeros((2, 2))

S[0,0] = np.dot(np.linalg.pinv(A), b)[0]

S[0,1] = np.dot(np.linalg.pinv(A), b)[1]

S[1,0] = np.dot(np.linalg.pinv(A), b)[1]

S[1,1] = 1

# get the SVD of S

U, D, V = np.linalg.svd(S)

D = np.sqrt(np.diag(D))

A = np.dot(np.dot(U, D), np.transpose(U))

H = np.zeros((3,3))

H[0,:] = [A[0,0], A[0,1], 0]

H[1,:] = [A[1,0], A[1,1], 0]

H[2,:] = [0, 0, 1]

return H

def build(L, M):

a = L[0]\*M[0]

b = (L[0]\*M[1]+L[1]\*M[0])/2

c = L[1]\*M[2]

d = (L[0]\*M[2]+L[2]\*M[0])/2

e = (L[1]\*M[2]+L[2]\*M[1])/2

result = [a,b,c,d,e]

return result

def get\_H\_one\_step(corners):

# calculate 5 pairs of orthogonal lines

l1 = np.cross(corners[0], corners[1])

l1 = l1 / np.max(l1)

m1 = np.cross(corners[1], corners[2])

m1 = m1 / np.max(m1)

l2 = np.cross(corners[1], corners[2])

l2 = l2 / np.max(l2)

m2 = np.cross(corners[2], corners[3])

m2 = m2 / np.max(m2)

l3 = np.cross(corners[2], corners[3])

l3 = l3 / np.max(l3)

m3 = np.cross(corners[3], corners[0])

m3 = m3 / np.max(m3)

l4 = np.cross(corners[3], corners[0])

l4 = l4 / np.max(l4)

m4 = np.cross(corners[0], corners[1])

m4 = m4 / np.max(m4)

l5 = np.cross(corners[4], corners[5])

l5 = l5 / np.max(l5)

m5 = np.cross(corners[5], corners[6])

m5 = m5 / np.max(m5)

A = np.zeros((5,5))

A[0,:] = build(l1, m1)

A[1,:] = build(l2, m2)

A[2,:] = build(l3, m3)

A[3,:] = build(l4, m4)

A[4,:] = build(l5, m5)

b = np.array([[-l1[2] \* m1[2]], [-l2[2] \* m2[2]], [-l3[2] \* m3[2]], [-l4[2] \* m4[2]], [-l5[2] \* m5[2]]])

# calculate the C matrix

ans = np.dot(np.linalg.pinv(A), b)

ans = ans / np.max(ans)

S = np.zeros((2, 2))

S[0,0] = ans[0]

S[0,1] = ans[1] / 2

S[1,0] = ans[1] / 2

S[1,1] = ans[2]

# get the SVD of S

U, D, V = np.linalg.svd(S)

# calculate the A matrix

D = np.sqrt(np.diag(D))

temp = np.dot(np.dot(U, D), np.transpose(U))

temp1 = np.array([ans[3]/2, ans[4]/2])

v = np.dot(np.linalg.pinv(temp), temp1)

H = np.zeros((3,3))

H[0,:] = [temp[0,0], temp[0,1], 0]

H[1,:] = [temp[1,0], temp[1,1], 0]

H[2,:] = [v[0], v[1], 1]

return H

###########Method 1 starts here############

# img 1

img1 = np.array([[642, 498], [642, 532], [666, 537], [666, 503]])

img1\_world = np.array([[0, 0], [0, 85], [75, 85], [75, 0]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img1.JPG"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H = get\_homography(img1\_world,img1)

img\_new = mapping(img\_target,H)

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img1\_method1.jpeg", img\_new)

# img 2

img2 = np.array([[480, 722], [481, 874], [606, 923], [600, 739]])

img2\_world = np.array([[0, 0], [0, 74], [84, 74], [84, 0]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img2.jpeg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H = get\_homography(img2\_world,img2)

img\_new = mapping(img\_target,H)

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img2\_method1.jpeg", img\_new)

# img 3

img3 = np.array([[2060, 700], [2092, 1483], [2695, 1333], [2666, 720]])

img3\_world = np.array([[0, 0], [0, 36], [55, 36], [55, 0]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img3.JPG"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H = get\_homography(img3\_world,img3)

img\_new = mapping(img\_target,H)

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img3\_method1.jpeg", img\_new)

# myimg 1

myimg1 = np.array([[597, 293], [589, 525], [822, 575], [820, 371]])

myimg1\_world = np.array([[0, 0], [0, 25], [32, 25], [32, 0]])

directory = "/home/xu1363/Documents/ECE 661/hw3/"

file\_target = "mypic1.jpg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H = get\_homography(myimg1\_world,myimg1)

img\_new = mapping(img\_target,H)

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg1\_method1.jpeg", img\_new)

# myimg 2

myimg2 = np.array([[466, 64], [329, 690], [770, 755], [839, 193]])

myimg2\_world = np.array([[0, 0], [0, 48], [33, 48], [33, 0]])

directory = "/home/xu1363/Documents/ECE 661/hw3/"

file\_target = "mypic2.jpg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H = get\_homography(myimg2\_world,myimg2)

img\_new = mapping(img\_target,H)

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg2\_method1.jpeg", img\_new)

###########Method 2 starts here############

# img 1

PQRS = np.array([[592,208],[585,522],[876,575],[864,297]])

PQRS\_HC = np.array([[592,208,1],[585,522,1],[876,575,1],[864,297,1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img1.JPG"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_projective(PQRS\_HC)

output1 = mapping(img\_target, H1)

plt.imshow(cv2.cvtColor(output1, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img1\_method2\_projective.jpeg", output1)

H2 = get\_affine(PQRS\_HC)

output2 = mapping(img\_target, np.dot(np.linalg.pinv(H2), H1))

plt.imshow(cv2.cvtColor(output2, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img1\_method2\_affine.jpeg", output2)

# img 2

PQRS = np.array([[367, 553], [362, 853], [641, 975], [621, 508]])

PQRS\_HC = np.array([[367, 553, 1], [362, 853, 1], [641, 975, 1], [621, 508, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img2.jpeg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_projective(PQRS\_HC)

output1 = mapping(img\_target, H1)

plt.imshow(cv2.cvtColor(output1, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img2\_method2\_projective.jpeg", output1)

H2 = get\_affine(PQRS\_HC)

output2 = mapping(img\_target, np.dot(np.linalg.pinv(H2), H1))

plt.imshow(cv2.cvtColor(output2, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img2\_method2\_affine.jpeg", output2)

# img 3

PQRS = np.array([[2087, 739], [2120, 1429], [2673, 1302], [2651, 749]])

PQRS\_HC = np.array([[2087, 739, 1], [2120, 1429, 1], [2673, 1302, 1], [2651, 749, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img3.JPG"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_projective(PQRS\_HC)

output1 = mapping(img\_target, H1)

plt.imshow(cv2.cvtColor(output1, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img3\_method2\_projective.jpeg", output1)

H2 = get\_affine(PQRS\_HC)

output2 = mapping(img\_target, np.dot(np.linalg.pinv(H2), H1))

plt.imshow(cv2.cvtColor(output2, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img3\_method2\_affine.jpeg", output2)

# myimg 1

PQRS = np.array([[597, 293], [589, 525], [822, 575], [820, 371]])

PQRS\_HC = np.array([[597, 293, 1], [589, 525, 1], [822, 575, 1], [820, 371, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/"

file\_target = "mypic1.jpg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_projective(PQRS\_HC)

output1 = mapping(img\_target, H1)

plt.imshow(cv2.cvtColor(output1, cv2.COLOR\_BGR2RGB))

plt.show()

plt.clf()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg1\_method2\_projective.jpeg", output1)

H2 = get\_affine(PQRS\_HC)

output2 = mapping(img\_target, np.dot(np.linalg.pinv(H2), H1))

plt.imshow(cv2.cvtColor(output2, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg1\_method2\_affine.jpeg", output2)

# myimg 2

PQRS = np.array([[466, 64], [329, 690], [770, 755], [839, 193]])

PQRS\_HC = np.array([[466, 64, 1], [329, 690, 1], [770, 755, 1], [839, 193, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/"

file\_target = "mypic2.jpg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_projective(PQRS\_HC)

output1 = mapping(img\_target, H1)

plt.imshow(cv2.cvtColor(output1, cv2.COLOR\_BGR2RGB))

plt.show()

plt.clf()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg2\_method2\_projective.jpeg", output1)

H2 = get\_affine(PQRS\_HC)

output2 = mapping(img\_target, np.dot(np.linalg.pinv(H2), H1))

plt.imshow(cv2.cvtColor(output2, cv2.COLOR\_BGR2RGB))

plt.show()

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg2\_method2\_affine.jpeg", output2)

###########Method 3 starts here############

# img 1

PQRS = np.array([[592,208],[585,522],[876,575],[864,297],[168,137],[115,587],[393,622]])

PQRS\_HC = np.array([[592,208,1],[585,522,1],[876,575,1],[864,297,1],[168,137,1],[115,587,1],[393,622,1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img1.JPG"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_H\_one\_step(PQRS\_HC)

img\_new = mapping(img\_target, np.linalg.inv(H1))

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img1\_method3.jpeg", img\_new)

# img 2

PQRS = np.array([[367, 553], [362, 853], [641, 975], [621, 508], [412, 100], [409, 374], [538, 311]])

PQRS\_HC = np.array([[367, 553, 1], [362, 853, 1], [641, 975, 1], [621, 508, 1], [412, 100, 1], [409, 374, 1], [538, 311, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img2.jpeg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_H\_one\_step(PQRS\_HC)

img\_new = mapping(img\_target, np.linalg.inv(H1))

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img2\_method3.jpeg", img\_new)

# img 3

PQRS = np.array([[2087, 739], [2120, 1429], [2673, 1302], [2651, 749], [746, 820], [790, 2002], [1749, 1577]])

PQRS\_HC = np.array([[2087, 739, 1], [2120, 1429, 1], [2673, 1302, 1], [2651, 749, 1], [746, 820, 1], [790, 2002, 1], [1749, 1577, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/hw3\_Task1\_Images/Images/"

file\_target = "Img3.JPG"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_H\_one\_step(PQRS\_HC)

img\_new = mapping(img\_target, np.linalg.inv(H1))

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"img3\_method3.jpeg", img\_new)

# myimg 1

PQRS = np.array([[597, 293], [589, 525], [822, 575], [820, 371], [1020, 442], [1032, 619], [1165, 648]])

PQRS\_HC = np.array([[597, 293, 1], [589, 525, 1], [822, 575, 1], [820, 371, 1], [1020, 442, 1], [1032, 619, 1], [1165, 648, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/"

file\_target = "mypic1.jpg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_H\_one\_step(PQRS\_HC)

img\_new = mapping(img\_target, np.linalg.inv(H1))

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg1\_method3.jpeg", img\_new)

# myimg 2

PQRS = np.array([[466, 64], [329, 690], [770, 755], [839, 193], [1053, 267], [1008, 799], [1332, 860]])

PQRS\_HC = np.array([[466, 64, 1], [329, 690, 1], [770, 755, 1], [839, 193, 1], [1053, 267, 1], [1008, 799, 1], [1332, 860, 1]])

directory = "/home/xu1363/Documents/ECE 661/hw3/"

file\_target = "mypic2.jpg"

img\_target = cv2.imread(directory+file\_target,cv2.IMREAD\_COLOR)

H1 = get\_H\_one\_step(PQRS\_HC)

img\_new = mapping(img\_target, np.linalg.inv(H1))

plt.imshow(cv2.cvtColor(img\_new, cv2.COLOR\_BGR2RGB))

cv2.imwrite("/home/xu1363/Documents/ECE 661/hw3/"+"myimg2\_method3.jpeg", img\_new)