ECE 661 – Homework 2

Ran Xu

xu943@purdue.edu

09/03/2018

1. Logic and math reductions

Given a point $P = (x_1, y_1)$ or $(x_1, y_1, 1)$ in homogeneous coordinate (HC) representation in real word plane, we use a homography H to map it to a point $P' = (x_1', y_1')$ in camera plane. Due to homogenous property, we assume H as follows,

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

Then we have the HC of P'

$$P' = HP = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11}x_1 + h_{12}y_1 + h_{13} \\ h_{21}x_1 + h_{22}y_1 + h_{23} \\ h_{31}x_1 + h_{32}y_1 + 1 \end{bmatrix} = (h_{31}x_1 + h_{32}y_1 + 1) \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix}$$

It can be represented in the following way,

$$\begin{bmatrix} x_1' \\ y_1' \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}$$

The h vector needs at least 4 pairs of points, a.k.a. 8 equations to be solved. However, more equations are also preferable.

Denote the i-th projection point $P_i' = (x_i', y_i')$, the h vector

$$h = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}]^T$$

and the i-th coefficient matrix

$$C_i = \begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_ix_i' & -y_ix_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_iy_i' & -y_iy_i' \end{bmatrix}$$

The solution with 4 pairs of projection points are shown as follows,

$$\begin{bmatrix} P_1' \\ P_2' \\ P_3' \\ P_4' \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} h, thush = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}^{-1} \begin{bmatrix} P_1' \\ P_2' \\ P_3' \\ P_4' \end{bmatrix}$$

Generally, the solution with n (n>4) pairs of projection points are shown as follows using generalized inverse,

$$\begin{bmatrix} P_1' \\ P_2' \\ \cdots \\ P_n' \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_n \end{bmatrix} h, thus\ h = \left(\begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_n \end{bmatrix}^T \begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_n \end{bmatrix} \right)^{-1} \begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_n \end{bmatrix}^T \begin{bmatrix} P_1' \\ P_2' \\ \cdots \\ P_n' \end{bmatrix}$$

2. Solution

2.1 Using given images Fig. 1(a)-(d).

2.1 (a) Map Fig. 1(d) to the frames in Fig. 1(a)-(c)

I use GIMP to find out the (x,y) coordinates of frames in Fig. 1(a)-(d) as shown in Table 1

Table *1*: frame coordinates in Fig. 1

	P (Left upper)	Q (Right upper)	R (Left bottom)	S (Right bottom)
Fig. 1(a)	(1467,75)	(2985,681)	(1440,2340)	(3048,2094)
Fig. 1(b)	(1278,258)	(3054,573)	(1245,2082)	(3081,1938)
Fig. 1(c)	(876,690)	(2866,300)	(856,2142)	(2922,2312)
Fig. 1(d)	(0,0)	(1279,0)	(0,719)	(1279,719)

Then using the equations in Sec. 1, I get the homographies between Fig. 1(a)-(c) and (d) as shown in Table 2.

Table 2: homographies between Fig. 1(a)-(c) and (d)

	From Fig. 1(d) to (a)	From Fig. 1(d) to (b)	From Fig. 1(d) to (c)
Homography	$\begin{bmatrix} 2.56 & -0.196 & 1.47 \times 10^3 \\ 0.787 & 2.89 & 75.0 \\ 4.61 \times 10^{-4} & -1.10 \times 10^{-4} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 2.18 & -0.111 & 1.28 \times 10^3 \\ 0.395 & 2.43 & 258 \\ 2.60 \times 10^{-4} & -5.22 \times 10^{-5} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.937 & -6.27 \times 10^{-2} & 876 \\ -0.370 & 1.93 & 690 \\ -2.16 \times 10^{-4} & -4.08 \times 10^{-5} & 1.00 \end{bmatrix}$

The resulting images are shown in Figure 1.

From Fig. 1(d) to (a) | From Fig. 1(d) to (b) | From Fig. 1(d) to (c)

Figure *1*: Resulting images mapping Fig. 1(d) to the frames in Fig. 1(a)-(c)

(b) Cascade homography

The homographies are summarized in Table 3. The product of the individual homogrphies is the same as the homography from Fig. 1(a) to (c).

Table 3: Homographies

	From Fig. 1(a) to (b)	From Fig. 1(b) to (c)	From Fig. 1(a) to (c)
Homography	$\begin{bmatrix} 0.718 & 1.73\times10^{-2} & 42.1 \\ -0.164 & 0.724 & 407 \\ -9.77\times10^{-5} & 1.59\times10^{-5} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.289 & 5.03\times10^{-6} & 290 \\ -0.330 & 0.601 & 786 \\ -1.93\times10^{-4} & 4.04\times10^{-8} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.181 & 9.69\times10^{-3} & 305 \\ -0.415 & 0.446 & 1.02\times10^{3} \\ -2.39\times10^{-4} & 1.27\times10^{-5} & 1.00 \end{bmatrix}$
	Product of the homography from Fig. 1(a) to (b) and that from Fig. 1(b) to (c)		
Homography	$\begin{bmatrix} 0.181 & 9.69\times10^{-3} & 305 \\ -0.415 & 0.446 & 1.02\times10^{3} \\ -2.39\times10^{-4} & 1.27\times10^{-5} & 1.00 \end{bmatrix}$		

By applying this homogrphy to Fig. 1(a), the resulting image is shown in Figure 2. The frame is very similar to that in Fig. 1 (c). Note that the origin in this figure is shifted to (304, -3420).

Figure *2*: Applying the homogrphy to Fig. 1(a)

2.1 Using own captured images

My own images are shown in Figure 3.

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

Figure 3: Own images

2.1 (a) Map Fig. 3(d) to the frames in Fig. 3(a)-(c)

I use GIMP to find out the (x,y) coordinates of frames in Fig. 3(a)-(d) as shown in Table 4.

Table 4: frame coordinates in Fig. 3

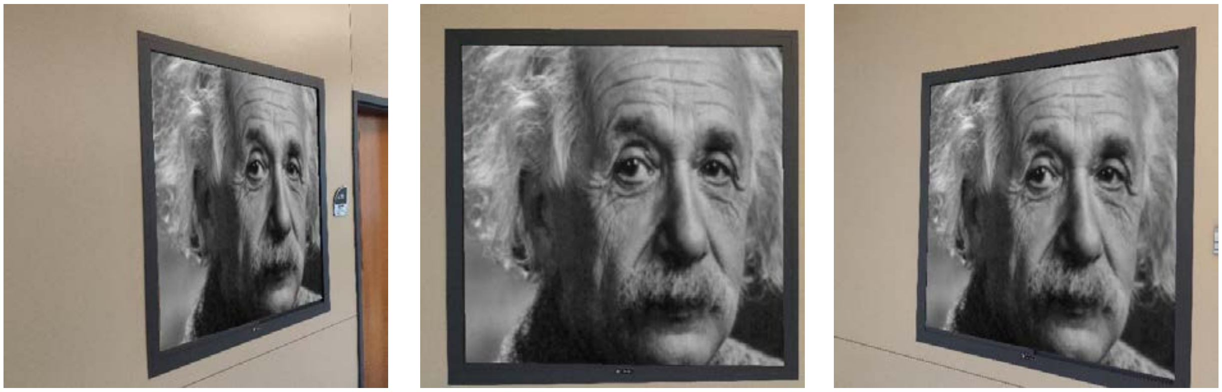
	P (Left upper)	Q (Right upper)	R (Left bottom)	S (Right bottom)
Fig. 3(a)	(192,62)	(406,110)	(205,450)	(413,383)
Fig. 3(b)	(54,54)	(470,56)	(59,466)	(472,470)
Fig. 3(c)	(126,107)	(447,58)	(119,418)	(442,488)
Fig. 3(d)	(0,0)	(879,0)	(0,719)	(879,719)

Then using the equations in Sec. 1, I get the homographies between Fig. 3(a)-(c) and (d) as shown in Table 5.

Table 5: homographies between Fig. 3(a)-(c) and (d)

	From Fig. 3(d) to (a)	From Fig. 3(d) to (b)	From Fig. 3(d) to (c)
Homography	$\begin{bmatrix} 0.440 & 2.22\times10^{-2} & 192 \\ 0.108 & 0.549 & 62.0 \\ 4.83\times10^{-4} & 2.02\times10^{-5} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.471 & 7.55\times10^{-3} & 54.0 \\ 1.96\times10^{-3} & 0.577 & 54.0 \\ -5.58\times10^{-6} & 1.01\times10^{-5} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.225 & -1.15\times10^{-2} & 126 \\ -7.39\times10^{-2} & 0.426 & 107 \\ -3.13\times10^{-4} & -1.45\times10^{-5} & 1.00 \end{bmatrix}$

The resulting images are shown in Figure 4.



From Fig. 3(d) to (a)   From Fig. 3(d) to (b)   From Fig. 3(d) to (c)

Figure 4: Resulting images mapping Fig. 1(d) to the frames in Fig. 1(a)-(c)

(b) Cascade homography

The homographies are summarized in Table 6. The product of the individual homogrphies is the same as the homography from Fig. 3(a) to (c).

Table 6: Homographies

	From Fig. 3(a) to (b)	From Fig. 3(b) to (c)	From Fig. 3(a) to (c)
Homography	$\begin{bmatrix} 1.01 & -2.47\times10^{-2} & -151 \\ -0.243 & 0.837 & 37.2 \\ -1.11\times10^{-3} & 2.28\times10^{-5} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.463 & -2.69\times10^{-2} & 97.9 \\ -0.153 & 0.712 & 73.0 \\ -6.29\times10^{-4} & -3.36\times10^{-5} & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.336 & -2.90\times10^{-2} & 24.8 \\ -0.375 & 0.551 & 112 \\ -1.60\times10^{-3} & 9.37\times10^{-6} & 1.00 \end{bmatrix}$
	Product of the homography from Fig. 3(a) to (b) and that from Fig. 3(b) to (c)		
Homography	$\begin{bmatrix} 0.336 & -2.90\times10^{-2} & 24.8 \\ -0.375 & 0.551 & 112 \\ -1.60\times10^{-3} & 9.37\times10^{-6} & 1.00 \end{bmatrix}$		

By applying this homogrphy to Fig. 3(a), the resulting image is shown in Figure 5. The frame is very similar to that in Fig. 3 (c). Note that the origin in this figure is shifted to (10, -367).

Figure 5: Applying the homogrphy to Fig. 3(a)

3. Source code

3.1 Helper functions that calculate the homography and map a region of an image to another image.

```
# This is the utility functions for ECE 661 hw2
import numpy as np
import copy
def Homography(A,B):
    # Find the homography from A to B, for example
    # A = np.array([[1467,75], [2985,681], [1440,2340], [3048,2094]])
    # B = np.array([[1278,258], [3054,573], [1245,2082], [3081,1938]])
    Projection = np.reshape(B, (B.shape[0]*B.shape[1],1))  #
(xB1,yB1,xB2,yB2,...)
    Coefficient = np.zeros((A.shape[0]*2, 8))  # number of points *2 x 8
    for index in range(A.shape[0]):
        Coefficient[index*2, 0] = A[index, 0]  # xA
        Coefficient[index*2, 1] = A[index, 1]  # yA
        Coefficient[index*2, 2] = 1
        Coefficient[index*2, 6] = - A[index, 0] * B[index, 0]  # -xA*xB
        Coefficient[index*2, 7] = - A[index, 1] * B[index, 0]  # -yA*xB
```

```
        Coefficient[index*2+1, 3] = A[index, 0] # xA
        Coefficient[index*2+1, 4] = A[index, 1] # yA
        Coefficient[index*2+1, 5] = 1
        Coefficient[index*2+1, 6] = - A[index, 0] * B[index, 1]  # -xA*yB
        Coefficient[index*2+1, 7] = - A[index, 1] * B[index, 1]  # -yA*yB
    h = np.matmul(np.linalg.inv(Coefficient), Projection)
    H_DA = np.array(([h[0][0], h[1][0], h[2][0]],
                     [h[3][0], h[4][0], h[5][0]],
                     [h[6][0], h[7][0], 1]))
    return H_DA

def Filling(imgA, A, imgB, B):
    # Fill in the area of A in imgA with the area of B in imgB
    # Note: B must be a rectangular area!
    # A = np.array([[1467,75], [2985,681], [1440,2340], [3048,2094]])
    # B = np.array([[0,0], [1280,0], [0,720], [1280,720]])
    # imgA could be a (2709, 3612, 3) numpy.ndarray
    # imgB could be a (720, 1280, 3) numpy.ndarray
    H_AB = Homography(A,B)

    # Construct a 3-D coordinate matrix of A
    # [ 0, 0, 0, ........., 1, 1, 1, ........., 2 ..]
    # [ 0, 1, 2, ..., 2708, 0, 1, 2, ..., 2708, ... ]
    # [ 1, 1, 1, 1, 1, 1, 1, ..................,1  ]
    # Meaning
    # [ X coordinate = Width direction, e.g. 3612]
    # [ Y coordinate = Height direction, e.g. 2709]
    # [ Ones ]
    HC_A = np.ones((3, imgA.shape[0]*imgA.shape[1])).astype(int)
    for indexH in range(imgA.shape[1]):
        HC_A[0, indexH*imgA.shape[0]:indexH*imgA.shape[0]+imgA.shape[0]] = \
            np.repeat([indexH], imgA.shape[0])
        HC_A[1, indexH*imgA.shape[0]:indexH*imgA.shape[0]+imgA.shape[0]] = \
            np.arange(imgA.shape[0])
    HC_MappedA = np.matmul(H_AB, HC_A)
    HC_MappedA = np.round(HC_MappedA/HC_MappedA[2,:]).astype(int)

    # Check what mapped cooredinates is inside B
    EditVector = np.logical_and(HC_MappedA[0,:]>=B[0,0],
HC_MappedA[0,:]<=B[1,0])
    EditVector = np.logical_and(HC_MappedA[1,:]>=B[0,1], EditVector)
    EditVector = np.logical_and(HC_MappedA[1,:]<=B[2,1], EditVector)

    # Refill the image
    # Map all pixels from imgA plane to imgB plane, replace the pixels that
mapped to B
    RefilledA_WithB = copy.deepcopy(imgA)
    for index in np.arange(imgA.shape[0]*imgA.shape[1])[EditVector]:
        RefilledA_WithB[HC_A[1,index], HC_A[0,index], :] = \
            imgB[HC_MappedA[1,index], HC_MappedA[0,index], :]
    return RefilledA_WithB

def Mapped(imgA, H):
    # Boundary mapping
    HC_Boundary_A = np.array([[0,imgA.shape[1]-1,0,imgA.shape[1]-1],
                              [0,0,imgA.shape[0]-1,imgA.shape[0]-1],
                              [1,1,1,1]]).astype(int)
    HC_Boundary_MappedA = np.matmul(H, HC_Boundary_A)
    HC_Boundary_MappedA =
(HC_Boundary_MappedA/HC_Boundary_MappedA[2,:]).astype(int)

    x_min = np.min(HC_Boundary_MappedA[0,:])
    x_max = np.max(HC_Boundary_MappedA[0,:])
```

```
    y_min = np.min(HC_Boundary_MappedA[1,:])
    y_max = np.max(HC_Boundary_MappedA[1,:])
    x_lim = x_max-x_min+1
    y_lim = y_max-y_min+1

    H_inverse = np.linalg.inv(H)
    # Construct a 3-D coordinate matrix "HC_mappedA" of mapped A (real HC)
    # [ 0, 0, 0, ........., 1, 1, 1, ........., (x_lim-1) ..] + x_min
    # [ 0, 1, 2, ..., (y_lim-1), 0, 1, 2, ..., (y_lim-1), ... ] + y_min
    # [ 1, 1, 1, 1, 1, 1, 1, ..................,1  ]
    # Meaning
    # [ X coordinate = Width direction, e.g. x_lim]
    # [ Y coordinate = Height direction, e.g. y_lim]
    # [ Ones ]
    HC_mappedA = np.ones((3, x_lim*y_lim)).astype(int)
    for indexH in range(x_lim):
        HC_mappedA[0, indexH*y_lim:indexH*y_lim+y_lim] = \
            np.repeat([indexH], y_lim) + x_min
        HC_mappedA[1, indexH*y_lim:indexH*y_lim+y_lim] = \
            np.arange(y_lim) + y_min
    HC_A = np.matmul(H_inverse, HC_mappedA)
    HC_A = (np.round(HC_A/HC_A[2,:])).astype(int)
    #print(np.max(HC_A[1,:]))

    A = np.array([[0,0], [imgA.shape[0],0],
                  [0,imgA.shape[1]], [imgA.shape[0],imgA.shape[1]]])
    # Check what mapped cooredinates is inside A
    EditVector = np.logical_and(HC_A[0,:]>=HC_Boundary_A[0,0],
HC_A[0,:]<=HC_Boundary_A[0,1])
    EditVector = np.logical_and(HC_A[1,:]>=HC_Boundary_A[1,0], EditVector)
    EditVector = np.logical_and(HC_A[1,:]<=HC_Boundary_A[1,2], EditVector)

    # Refill the image
    # Map all pixels inversely from mappedA plane to imgA plane, replace the
pixels that mapped to A
    mappedA = np.zeros((y_lim,x_lim,3)).astype(int)
    for index in np.arange(x_lim*y_lim)[EditVector]:
        mappedA[HC_mappedA[1,index]-y_min, HC_mappedA[0,index]-x_min, :] = \
            imgA[HC_A[1,index], HC_A[0,index], :]
    return (x_min,y_min,mappedA)
```

3.2 Main function of task 1

```
# This is the main python script for ECE 661 hw2
# To run: python3 main.py
import numpy as np
from PIL import Image
from homography_util import *

A = np.array([[1467,75], [2985,681], [1440,2340], [3048,2094]])
B = np.array([[1278,258], [3054,573], [1245,2082], [3081,1938]])
C = np.array([[876,690], [2866,300], [856,2142], [2922,2312]])
D = np.array([[0,0], [1279,0], [0,719], [1279,719]])
imgA = np.array(Image.open("PicsHw2/1.jpg"))
imgB = np.array(Image.open("PicsHw2/2.jpg"))
imgC = np.array(Image.open("PicsHw2/3.jpg"))
imgD = np.array(Image.open("PicsHw2/Jackie.jpg"))

# Task 1a
# Hymnographies and resulting images from Fig. 1(d) to Fig. 1(a)
H_DA = Homography(D,A)
print("H_DA = ", H_DA)
RefilledA_WithD = Filling(imgA, A, imgD, D)
```

```
RefilledImgA_WithD = Image.fromarray(RefilledA_WithD, 'RGB')
RefilledImgA_WithD.save("Task1a1.jpg")

# Homographies and resulting images from Fig. 1(d) to Fig. 1(b)
H_DB = Homography(D,B)
print("H_DB = ", H_DB)
RefilledB_WithD = Filling(imgB, B, imgD, D)
RefilledImgB_WithD = Image.fromarray(RefilledB_WithD, 'RGB')
RefilledImgB_WithD.save("Task1a2.jpg")

# Homographies and resulting images from Fig. 1(d) to Fig. 1(c)
H_DC = Homography(D,C)
print("H_DC = ", H_DC)
RefilledC_WithD = Filling(imgC, C, imgD, D)
RefilledImgC_WithD = Image.fromarray(RefilledC_WithD, 'RGB')
RefilledImgC_WithD.save("Task1a3.jpg")
print("----------")

# Task 1b
# Homographies from Fig. 1(a) to Fig. 1(b)
H_AB = Homography(A,B)
print("H_AB = ", H_AB)

# Homographies from Fig. 1(b) to Fig. 1(c)
H_BC = Homography(B,C)
print("H_BC = ", H_BC)

# Homographies from Fig. 1(a) to Fig. 1(c)
H_AC = Homography(A,C)
print("H_AC = ", H_AC)

H_ABC = np.matmul(H_BC, H_AB)
H_ABC = H_ABC/H_ABC[2][2]
print("H_ABC = ", H_ABC)

# Mapping the imgA to imgC plane
(x_min, y_min, mappedA) = Mapped(imgA, H_AC)
print("(x_min, y_min) = (%d, %d)" %(x_min, y_min))
mappedimgA = Image.fromarray(np.uint8(mappedA), 'RGB')
mappedimgA.save("Task1b.jpg")
```

3.3 Main function of task 2

```
# This is the main python script for ECE 661 hw2
# To run: python3 main2.py
import numpy as np
from PIL import Image
from homography_util import *

A = np.array([[192,62], [406,110], [205,450], [413,383]])
B = np.array([[54,54], [470,56], [59,466], [472,470]])
C = np.array([[126,107], [447,58], [119,418], [442,488]])
D = np.array([[0,0], [879,0], [0,719], [879,719]])
imgA = np.array(Image.open("PicsHw2/2-1.jpg"))  # 500W x 500H image
imgB = np.array(Image.open("PicsHw2/2-2.jpg"))  # 500W x 500H image
imgC = np.array(Image.open("PicsHw2/2-3.jpg"))  # 500W x 500H image
imgD = np.array(Image.open("PicsHw2/2-4.jpg"))  # 880W x 720H image

# Task 1a
# Hymnographies and resulting images from Fig. 1(d) to Fig. 1(a)
H_DA = Homography(D,A)
print("H_DA = ", H_DA)
RefilledA_WithD = Filling(imgA, A, imgD, D)
```

```
RefilledImgA_WithD = Image.fromarray(RefilledA_WithD, 'RGB')
RefilledImgA_WithD.save("Task2a1.jpg")

# Homographies and resulting images from Fig. 1(d) to Fig. 1(b)
H_DB = Homography(D,B)
print("H_DB = ", H_DB)
RefilledB_WithD = Filling(imgB, B, imgD, D)
RefilledImgB_WithD = Image.fromarray(RefilledB_WithD, 'RGB')
RefilledImgB_WithD.save("Task2a2.jpg")

# Homographies and resulting images from Fig. 1(d) to Fig. 1(c)
H_DC = Homography(D,C)
print("H_DC = ", H_DC)
RefilledC_WithD = Filling(imgC, C, imgD, D)
RefilledImgC_WithD = Image.fromarray(RefilledC_WithD, 'RGB')
RefilledImgC_WithD.save("Task2a3.jpg")
print("----------")

# Task 1b
# Homographies from Fig. 1(a) to Fig. 1(b)
H_AB = Homography(A,B)
print("H_AB = ", H_AB)

# Homographies from Fig. 1(b) to Fig. 1(c)
H_BC = Homography(B,C)
print("H_BC = ", H_BC)

# Homographies from Fig. 1(a) to Fig. 1(c)
H_AC = Homography(A,C)
print("H_AC = ", H_AC)

H_ABC = np.matmul(H_BC, H_AB)
H_ABC = H_ABC/H_ABC[2][2]
print("H_ABC = ", H_ABC)

# Mapping the imgA to imgC plane
(x_min, y_min, mappedA) = Mapped(imgA, H_AC)
print("(x_min, y_min) = (%d, %d)" %(x_min, y_min))
mappedimgA = Image.fromarray(np.uint8(mappedA), 'RGB')
mappedimgA.save("Task2b.jpg")
```