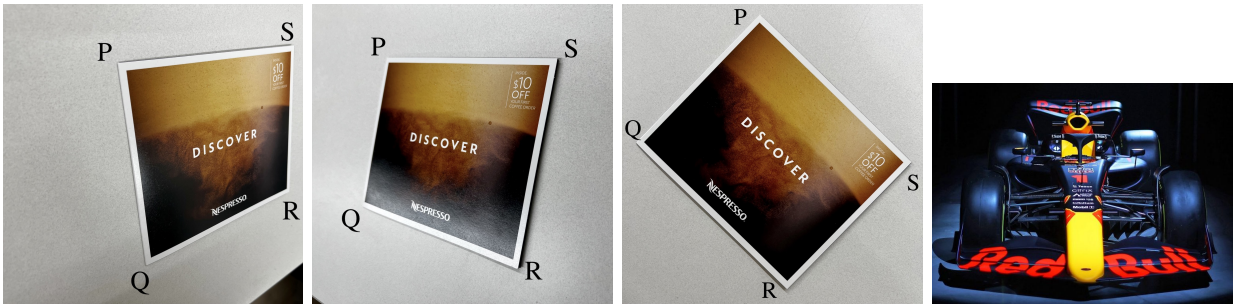
Task 1

Task 1.1

Using just four points, pick a region of interest (ROI) from the car image in Fig. 1d and project that region on the PQRS frames shown in Figs. 1a, 1b, and 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) images shown in Figs. 1d and 1b, and (3) images shown in Figs. 1d and 1c.



(a) 1a (b) 1b (c) 1c (d) 1d

Figure 1: Input Images

Background

Brief Summary of the process:

1. Get coordinates of the 4 points corresponding to PQRS corners for each frame shown in Figs. 1a, 1b, 1c.
2. Get 4 coordinates of ROI from the car image in Fig. 1d
3. Create a box using the coordinates on each image.
4. Create a mask of the frames onto which the car image will be projected.
5. Calculate the Homographies between each pair of images in Figs. 1d and 1a, Figs. 1d and 1b, and Figs. 1d and 1c. The car image is the domain space and the frame image is the range space.
6. Calculate H^{-1} to multiply it with pixel on the frame image to find the corresponding pixel on the car image.
7. Loop through each pixel on the frame image and copy the color of the corresponding pixel of the car image on to the frames.

Getting the coordinates: I used an online tool (https://pixspy.com/). It allowed me to upload a picture, and get the coordinates of the pixels when I hover the mouse above them.

The coordinates of the pixels for each fig. are shown below:

	Fig. 1a	Fig. 1b	Fig. 1c		Fig. 1d
P	[488, 252]	[319, 228]	[584, 48]	P'	[488, 252]
Q	[612, 1112]	[210, 860]	[62, 590]	Q'	[612, 1112]
R	[1222, 798]	[874, 1128]	[702, 1213]	R'	[1222, 798]
S	[1242, 177]	[1042, 229]	[1228, 674]	S'	[1242, 177]

Table 1

ROI of car image: For this task I chose the whole image since it was easier to get the coordinates for the corners.

Creating a box using the coordinates: For the frame images, the coordinates of PQRS corners allowed to me to draw a box by connecting the points with a line using the OpenCV tool 'polylines'. This allowed me to see if the point coordinates that I got from the online tool were correct or not. The output images Fig. 2a, 2b, and 2c with the box around the PQRS region is shown in the Output section below.

Creating a mask of the frames: After creating a box around the PQRS region on each frame image, I made a copy of the image and then I changed all the pixels inside the box to be black in color using the OpenCV tool 'fillPoly' . This allowed me to distinguish pixels inside the ROI with the pixels outside the ROI when looping through all the pixels and copying the pixel colors from the car image to the frame image. Figs. 3a, 3b, and 3c with the mask on the frame are shown below in the Output section.

Calculating Homographies: To calculate the Homography between a point x in the domain space and a point x' in the range space, the following matrix equation was simplified and solved simultaneously:

$$x' = \mathrm{H}x$$

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$x'_1 = h_{11}x_1 + h_{12}x_2 + h_{13}x_3$$

$$x'_2 = h_{21}x_1 + h_{22}x_2 + h_{23}x_3$$

$$x'_3 = h_{31}x_1 + h_{32}x_2 + x_3$$

Dividing the first two equations by the third equation

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

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

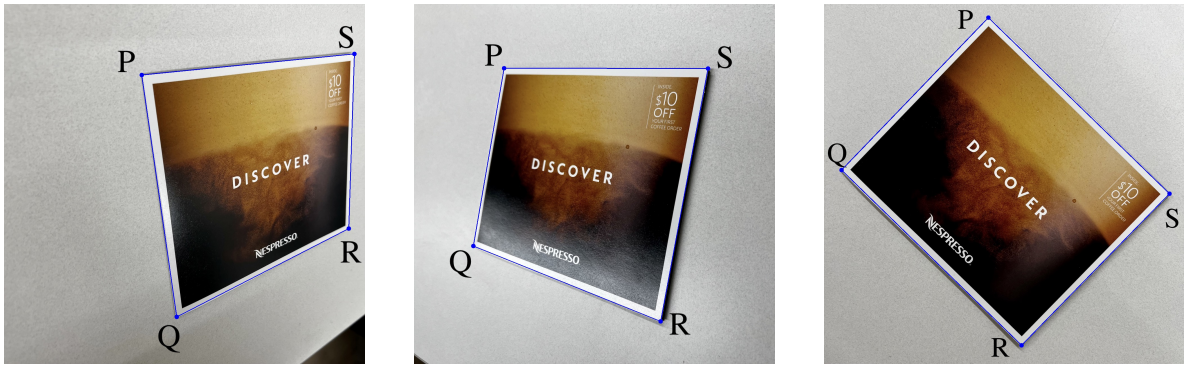
Since we have 8 unknowns and each given pair of coordinates give 2 equations, 4 pairs of coordinates are used to form 8 equations and this is simplified into this matrix form:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_1 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}$$

Showing this as $Ax = b$, x can be calculated as $x = A^{-1}b$.

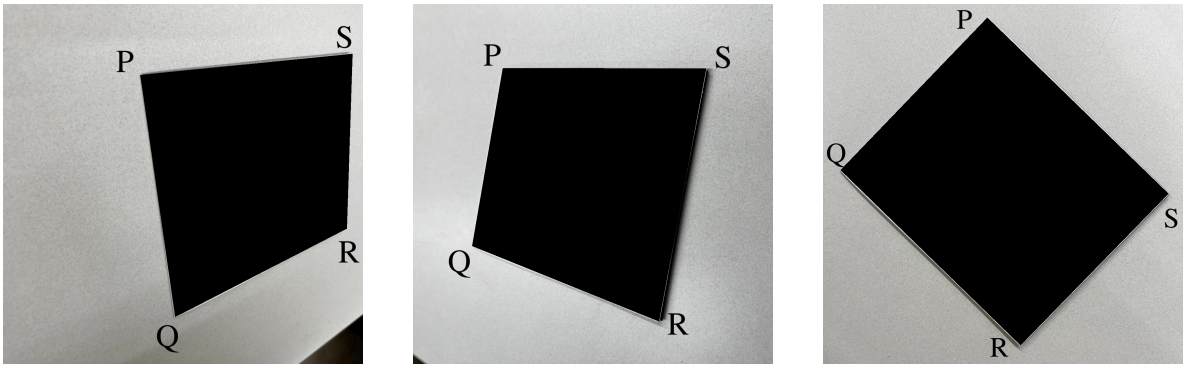
Outputs

The output images for task 1.1:



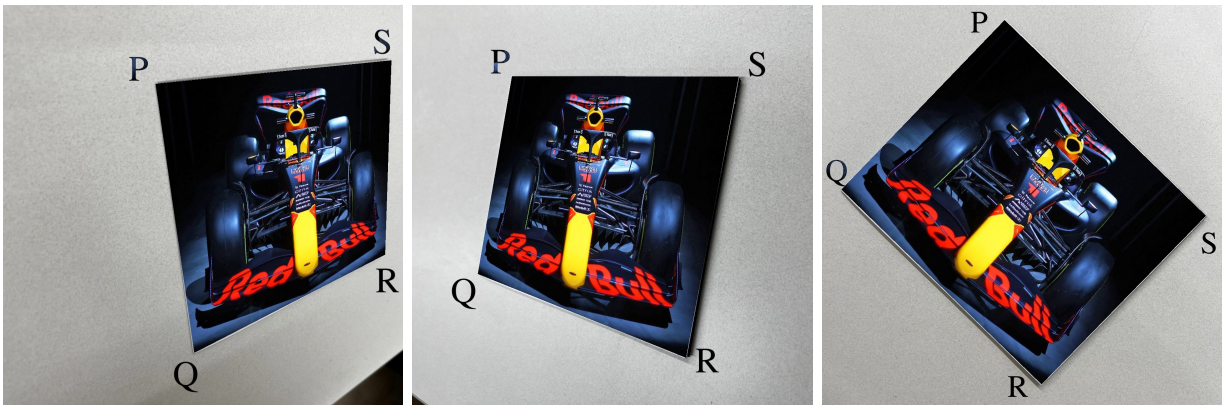
(a) Fig. 2a (b) Fig. 2b (c) Fig. 2c

Figure 2: Frame images with box drawn around the PQRS region



(a) Fig. 3a (b) Fig. 3b (c) Fig. 3c

Figure 3: Frame images with mask drawn on the PQRS region



(a) Fig. 4a (b) Fig. 4b (c) Fig. 4c

Figure 4: Car Image mapped on Frame images

Task 1.2

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

Background

Brief Summary of the process:

1. Get coordinates of the 4 points corresponding to PQRS corners for each frame shown in Figs. 1a, 1b, 1c.
2. Calculate the Homographies between images shown in Figs. 1a and 1b, and between images shown in Figs. 1b and 1c.
3. Loop through each pixel on Fig 1c and apply the inverse of the product of the two homographies on each pixel to find the corresponding pixel on image 1a and copy the colors.

Calculating Homographies: The homographies are calculated in the same way as explained under Task 1.1. The homography calculated between Figs. 1a and 1b is H_{ab} and the homography calculated between Figs. 1b and 1c is H_{bc}. The inverse of the product of homographies is H_{ac}^{-1}.

Outputs

Here is the output for task 1.2:

Fig. 5a is the same as fig. 1a, Fig. 5b is the same as Fig. 1c, and Fig. 5c is the projected image of 1a onto 1c.

(a) Fig. 5a: 1a (b) Fig. 5b: 1c (c) Fig. 5c: 1a mapped onto 1c

Figure 5: Image 1a mapped on to 1c

Task 1.3

Using only affine homographies, map the entire car image to the PQRS regions in all three card images. Report the results. If one of the results is better than the others, explain why.

Background

Brief Summary of the process:

1. Get coordinates of the 4 points corresponding to PQRS corners for each frame shown in Figs. 1a, 1b, 1c.
2. Get 4 coordinates of ROI from the car image in Fig. 1d
3. Create a box using the coordinates on each image.
4. Create a mask of the frames onto which the car image will be projected.
5. Calculate the Affine Homographies between each pair of images in Figs. 1d and 1a, Figs. 1d and 1b, and Figs. 1d and 1c. The car image is the domain space and the frame image is the range space. ’
6. Calculate H^{-1} to multiply it with pixel on the frame image to find the corresponding pixel on the car image.
7. Loop through each pixel on the mask image of the frames and copy the color of the corresponding pixel of the car image on to the frames.

Calculating Homographies: To calculate the affine Homography between a point x in the domain space and a point x' in the range space, the following matrix equation was simplified and solved simultaneously:

$$x' = \mathrm{H}x$$

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$x'_1 = h_{11}x_1 + h_{12}x_2 + h_{13}x_3$$

$$x'_2 = h_{21}x_1 + h_{22}x_2 + h_{23}x_3$$

$$x'_3 = x_3$$

Dividing the first two equations by the third equation

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

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

Since we have 6 unknowns and each given pair of coordinates give 2 equations, minimum of 3 pairs of coordinates are required to form at least 6 equations. To use all the four corners of the frame images, 4 pairs of coordinates are used and this is simplified into this matrix form:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & 0 & 0 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_4 & y_1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$$

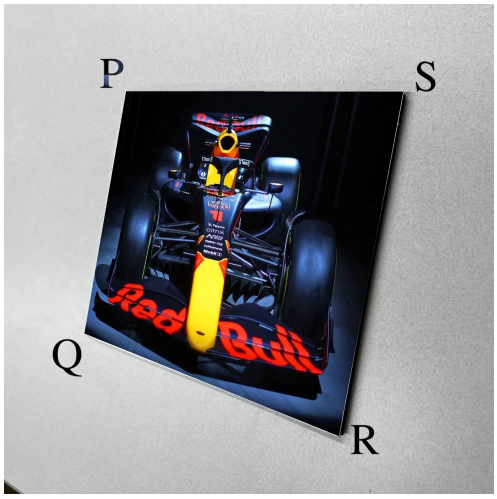
Showing this as $Ax = b$, x can be calculated as $x = A^{-1}b$.

Outputs

The output images for task 1.3:
As can be seen from the images, the projection of the car image onto 1a and 1b frame images were not perfect. The sides of 1a and 1b frame images are not parallel to each other but the parallel lines in the projection are intact because of affine transformation and this can be seen at the sides of the frames. The projection was near to perfect on 1c since all the sides of the frames are parallel to each other so the image projection was able to fit perfectly. The images of Projective transformation and Affine transformation are placed next to each other for comparison.

(a) Projective on Card 1

(b) Affine on Card 1



(c) Projective on Card 2

(d) Affine on Card 2

(e) Projective on Card 3

(f) Affine on Card 3

Figure 6: Car image projected onto 1a using protective and affine transformation

Task 2

Repeat the steps of Task 1 using your own images. You can capture three images of a planar surface from three different viewpoints such as the ones shown in Figs. 1a, 1b, and 1c. For the fourth image you can obtain a picture of your choice (animal, celebrity, etc.) from the Internet or use a picture of your own.
For this task, I chose the following images:

(a) 7a (b) 7b (c) 7c (d) 7d

Figure 7: Input Images

Task 2.1

Using four points, the AmongUs image was projected onto the frames shown above. Same methods were used as before. The only changes were the point coordinates and the images.

Output

(a) Fig. 8a (b) Fig. 8b (c) Fig. 8c

Figure 8: Frame images with mask drawn on the ROI

(a) Fig. 9a (b) Fig. 9b (c) Fig. 9c

Figure 9: AmongUs Image mapped on Frame images

Task 2.2

Using four points of each frame image, their homographies were calculated and then multiplied. The product of the homographies was applied on 7a and projected onto 7c. Same methods were used as before. The only changes were the point coordinates and the images.

Output

(a) Fig. 10a (b) Fig. 10b (c) Fig. 10c

Figure 10: Frame images with mask drawn on the ROI

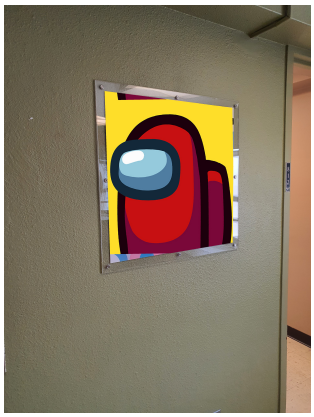
Task 2.3

Using only affine homographies, the AmongUs image was projected onto the frame images. Same methods were used as before. The only changes were the point coordinates and the images.As seen from the images, the only near perfect projection is onto 7c which has parallel

sides. The other two frames don't have parallel sides and so the projection doesn't fit properly.

Output

(a) Projective on Card 1



(b) Affine on Card 1

(c) Projective on Card 2

(d) Affine on Card 2

(e) Projective on Card 3

(f) Affine on Card 3

Figure 11: Car image projected onto 1a using protective and affine transformation

Code

```
from pydoc import getpager
from xml import dom
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from PIL import Image

#-------------------------Functions-------------------------------#
def drawCircleNBox(img, coords):
    img_box = img.copy()
    #draw circles at the coords
    for x,y in coords:
        cv2.circle(img_box, (x,y), 8, (255, 0, 0), -1)

    #draw lines connecting the coords to form a box
    coords.append(coords[0])
    cv2.polylines(img_box, [np.array(coords)], True, (255, 0, 0), 2)

    #return Img
    return img_box

def getMask(img, coords):
    img_mask = img.copy()
    img_mask = cv2.fillPoly(img_mask, [np.array(coords)], (0, 0, 0))
    return img_mask

def calcHomography(dom_coords, range_coords):
    ## AH = b

    # A and B Matrix
    A = np.zeros((8,8))
    b = np.zeros((8,1))
    for i in range(4):
        xp = range_coords[i][0]
        yp = range_coords[i][1]
        x = dom_coords[i][0]
        y = dom_coords[i][1]
        # A matrix
        A[2*i][0:8] = [x, y, 1 , 0 , 0 , 0 ,-x*xp, -y*xp]
        A[2*i+1][0:8] = [0, 0, 0, x, y, 1, -x*yp, -y*yp]
        # B matrix
        b[2*i] = xp
        b[2*i+1] = yp

    # H matrix
    H = np.zeros((3,3))
    H = np.matmul(np.linalg.pinv(A),b)
    H = np.append(H,[1])
    H = np.reshape(H, (3,3))
    return H
```

```
def calcAffineHomography(dom_coords, range_coords):
    ## AH = b

    # A and B Matrix
    A = np.zeros((8,8))
    b = np.zeros((8,1))
    for i in range(4):
        xp = range_coords[i][0]
        yp = range_coords[i][1]
        x = dom_coords[i][0]
        y = dom_coords[i][1]
        # A matrix
        A[2*i][0:8] = [x, y, 1 , 0 , 0 , 0 ,0, 0]
        A[2*i+1][0:8] = [0, 0, 0, x, y, 1, 0, 0]
        # B matrix
        b[2*i] = xp
        b[2*i+1] = yp

    # H matrix
    H = np.zeros((3,3))
    H = np.matmul(np.linalg.pinv(A),b)
    H = np.append(H,[1])
    H = np.reshape(H, (3,3))
    return H


def mappingHom(img_x, img_xp, mask_img, H):
    img_mapped = img_xp.copy()
    for row in range(img_xp.shape[1]):
        for col in range(img_xp.shape[0]):
            if any(mask_img[col][row])==0:
                domain_coord = np.array((row, col, 1))
                range_coord = np.matmul(H,domain_coord)
                range_coord = range_coord / range_coord[2]
                range_coord = range_coord.astype(int)
                if (range_coord[0]<img_x.shape[1] and range_coord[1]<img_x
                                                        .shape[0]):
                    img_mapped[col, row] = img_x[range_coord[1],
                                                    range_coord[0]]
    return img_mapped

def getMappedImg(dom_img, dom_img_coords, range_img, range_img_coords):
    # draw circles and bounding box on card
    boxImg = drawCircleNBox(range_img, range_img_coords)

    # Create a mask image
    range_img_mask = getMask(range_img, range_img_coords)

    # Generate a Homography Matrix
    H = calcHomography(dom_img_coords, range_img_coords)
    #Inverse the H to go from range to domain
    H = np.linalg.pinv(H)

    # Map points from domain to range
```

```
    mappedImg = mappingHom(dom_img, range_img, range_img_mask, H)

    return boxImg, range_img_mask, mappedImg

def getMappedImg_affine(dom_img, dom_img_coords, range_img,
                                    range_img_coords):
    # draw circles and bounding box on card
    boxImg = drawCircleNBox(range_img, range_img_coords)

    # Create a mask image
    range_img_mask = getMask(range_img, range_img_coords)

    # Generate a Homography Matrix
    H = calcAffineHomography(dom_img_coords, range_img_coords)
    #Inverse the H to go from range to domain
    H = np.linalg.pinv(H)

    # Map points from domain to range
    mappedImg = mappingHom(dom_img, range_img, range_img_mask, H)

    return boxImg, range_img_mask, mappedImg

# ------------------------------Task 1.1-------------------------------#

# Image Coordinates
car_coords = [(0,0), (0, 558), (760, 558), (760,0)]
card1_coords = [(488, 252), (612, 1112),(1222, 798), (1242, 177)]
card2_coords = [(319, 228), (210, 860), (874, 1128), (1042, 229)]
card3_coords = [(584, 48), (62, 590), (702, 1213), (1228, 674)]

# Input Images
car = cv2.imread('car.jpg')
img_card1 = cv2.imread('card1.jpeg')
img_card2 = cv2.imread('card2.jpeg')
img_card3 = cv2.imread('card3.jpeg')

#----------Projecting car on card 1-----------------#
card1_boxImg, card1_mask, card1_mapped = getMappedImg(car, car_coords,
                                    img_card1, card1_coords)

#----------Projecting car on card 2-----------------#
card2_boxImg, card2_mask, card2_mapped = getMappedImg(car, car_coords,
                                    img_card2, card2_coords)

#----------Projecting car on card 3-----------------#
card3_boxImg, card3_mask, card3_mapped = getMappedImg(car, car_coords,
                                    img_card3, card3_coords)


# Save images
cv2.imwrite('card1_boxed.jpeg',card1_boxImg)
cv2.imwrite('card1_mask.jpeg', card1_mask)
cv2.imwrite('card1_mapped.jpeg',card1_mapped)
```

```
cv2.imwrite('card2_boxed.jpeg',card2_boxImg)
cv2.imwrite('card2_mask.jpeg', card2_mask)
cv2.imwrite('card2_mapped.jpeg',card2_mapped)

cv2.imwrite('card3_boxed.jpeg',card3_boxImg)
cv2.imwrite('card3_mask.jpeg', card3_mask)
cv2.imwrite('card3_mapped.jpeg',card3_mapped)



# ------------------------------Task 1.2-------------------------------#
# Calculating H_ab
H_ab = calcHomography(card1_coords, card2_coords)

# Calculating H_bc
H_bc = calcHomography(card2_coords, card3_coords)

# Calculating H_ac
H_ac = np.matmul(H_ab, H_bc)

# Calculating H_ac inv
H_ac_inv = np.linalg.inv(H_ac)

# Mapping 1a to 1c
mappedImg3 = np.zeros(img_card3.shape, dtype='uint8')
for row in range(img_card3.shape[1]):
    for col in range(img_card3.shape[0]):
        domain_coord = np.array((row, col, 1))
        range_coord = np.matmul(H_ac_inv,domain_coord)
        range_coord = range_coord / range_coord[2]
        range_coord = range_coord.astype(int)
        if (range_coord[0]<img_card1.shape[1] and range_coord[1]<img_card1
                                            .shape[0] and range_coord[0]>
                                            0 and range_coord[1]>0):
            mappedImg3[col, row] = img_card1[range_coord[1], range_coord[0
                                                ]]

# Save image
cv2.imwrite('card1_3.jpeg',mappedImg3)

#------------------------------Task 1.3-------------------------------#
# Map the images
card1_boxImg_affine, card1_mask_affine, card1_mapped_affine =
                                    getMappedImg_affine(car, car_coords,
                                    img_card1, card1_coords)
card2_boxImg_affine, card2_mask_affine, card2_mapped_affine =
                                    getMappedImg_affine(car, car_coords,
                                    img_card2, card2_coords)
card3_boxImg_affine, card3_mask_affine, card3_mapped_affine =
                                    getMappedImg_affine(car, car_coords,
                                    img_card3, card3_coords)

# Save images
cv2.imwrite('card1_boxed_affine.jpeg',card1_boxImg_affine)
```

```
cv2.imwrite('card1_mask_affine.jpeg', card1_mask_affine)
cv2.imwrite('card1_mapped_affine.jpeg',card1_mapped_affine)

cv2.imwrite('card2_boxed_affine.jpeg',card2_boxImg_affine)
cv2.imwrite('card2_mask_affine.jpeg', card2_mask_affine)
cv2.imwrite('card2_mapped_affine.jpeg',card2_mapped_affine)

cv2.imwrite('card3_boxed_affine.jpeg',card3_boxImg_affine)
cv2.imwrite('card3_mask_affine.jpeg', card3_mask_affine)
cv2.imwrite('card3_mapped_affine.jpeg',card3_mapped_affine)

#---------------------------------------Task 2.1
                                      ----------------------------------------#


# Image Coordinates
amongus_coords = [(0,0), (0,511), (511,511), (511, 0)]
frame1_coords = [(991,850), (1047,2553), (2253, 2347), (2261, 1017)]
frame2_coords = [(950,1245), (968, 2520), (2045, 2774), (2045, 1003)]
frame3_coords = [(590, 943), (640, 2691), (2349, 2681), (2370, 932)]

# Input Images
amongus_img = cv2.imread('amongus.jpg')
frame1_img = cv2.imread('frame1.jpg')
frame2_img = cv2.imread('frame2.jpg')
frame3_img = cv2.imread('frame3.jpg')

#----------Projecting amongUs on frame 1-----------------#
frame1_boxImg, frame1_mask, frame1_mapped = getMappedImg(amongus_img,
                                    amongus_coords, frame1_img,
                                    frame1_coords)

#----------Projecting amongUs on frame 2-----------------#
frame2_boxImg, frame2_mask, frame2_mapped = getMappedImg(amongus_img,
                                    amongus_coords, frame2_img,
                                    frame2_coords)

#----------Projecting amongUs on frame 3-----------------#
frame3_boxImg, frame3_mask, frame3_mapped = getMappedImg(amongus_img,
                                    amongus_coords, frame3_img,
                                    frame3_coords)


# # Save images
cv2.imwrite('frame1_boxImg.jpeg',frame1_boxImg)
cv2.imwrite('frame1_mask.jpeg', frame1_mask)
cv2.imwrite('frame1_mapped.jpeg',frame1_mapped)

cv2.imwrite('frame2_boxImg.jpeg',frame2_boxImg)
cv2.imwrite('frame2_mask.jpeg', frame2_mask)
cv2.imwrite('frame2_mapped.jpeg',frame2_mapped)

cv2.imwrite('frame3_boxImg.jpeg',frame3_boxImg)
cv2.imwrite('frame3_mask.jpeg', frame3_mask)
```

```
cv2.imwrite('frame3_mapped.jpeg',frame3_mapped)

# ----------------------------Task 2.2-----------------------------#
# Calculating H_ab
H_ab = calcHomography(frame1_coords, frame2_coords)

# Calculating H_bc
H_bc = calcHomography(frame2_coords, frame3_coords)

# Calculating H_ac
H_ac = np.matmul(H_ab, H_bc)

# Calculating H_ac inv
H_ac_inv = np.linalg.inv(H_ac)

# Mapping 7a to 7c
mappedFrame = np.zeros(frame3_img.shape, dtype='uint8')
for row in range(frame3_img.shape[1]):
    for col in range(frame3_img.shape[0]):
        domain_coord = np.array((row, col, 1))
        range_coord = np.matmul(H_ac_inv,domain_coord)
        range_coord = range_coord / range_coord[2]
        range_coord = range_coord.astype(int)
        if (range_coord[0]<frame1_img.shape[1] and range_coord[1]<
                                            frame1_img.shape[0] and
                                            range_coord[0]>0 and
                                            range_coord[1]>0):
            mappedFrame[col, row] = frame1_img[range_coord[1], range_coord
                                                [0]]

# Save image
cv2.imwrite('frame1_3.jpeg',mappedFrame)

#----------------------------Task 2.3-----------------------------#
# Map the images
frame1_boxImg_affine, frame1_mask_affine, frame1_mapped_affine =
                                    getMappedImg_affine(amongus_img,
                                    amongus_coords, frame1_img,
                                    frame1_coords)
frame2_boxImg_affine, frame2_mask_affine, frame2_mapped_affine =
                                    getMappedImg_affine(amongus_img,
                                    amongus_coords, frame2_img,
                                    frame2_coords)
frame3_boxImg_affine, frame3_mask_affine, frame3_mapped_affine =
                                    getMappedImg_affine(amongus_img,
                                    amongus_coords, frame3_img,
                                    frame3_coords)

cv2.imwrite('frame1_boxed_affine.jpeg',frame1_boxImg_affine)
cv2.imwrite('frame1_mask_affine.jpeg', frame1_mask_affine)
cv2.imwrite('frame1_mapped_affine.jpeg',frame1_mapped_affine)

cv2.imwrite('frame2_boxed_affine.jpeg',frame2_boxImg_affine)
cv2.imwrite('frame2_mask_affine.jpeg', frame2_mask_affine)
```

```
cv2.imwrite('frame2_mapped_affine.jpeg',frame2_mapped_affine)

cv2.imwrite('frame3_boxed_affine.jpeg',frame3_boxImg_affine)
cv2.imwrite('frame3_mask_affine.jpeg', frame3_mask_affine)
cv2.imwrite('frame3_mapped_affine.jpeg',frame3_mapped_affine)
```