ECE661 HW2

Yueru Duan duan99@purdue.edu

Task 1

Logic - Find Homography Matrix

The linear transformation that maps point $\mathbf{x} = (x, y)$ ($\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ in $\mathbb{R}^3$, where $x = \frac{x_1}{x_3}$ and $y = \frac{x_2}{x_3}$) in the domain (which is the car image in this case) to point $\mathbf{x}' = (x', y')$ ($\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$ in $\mathbb{R}^3$, where $x' = \frac{x'_1}{x'_3}$ and $y' = \frac{x'_2}{x'_3}$) in the range (which is card1, card2 and card3 images respectively) is given by:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

where $\mathbf{H}$ is a non-singular 3×3 homogeneous matrix.

$$\mathbf{H} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & 1 \end{pmatrix}$$

The entry at the right bottom is 1 since it is only the ratios of the elements of $\mathbf{H}$ that matter. Thus, we have

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Expanding the above equation, we get

$$
\begin{aligned}
x'_1 &= a_{11}x_1 + a_{12}x_2 + t_x x_3 \\
x'_2 &= a_{21}x_1 + a_{22}x_2 + t_y x_3 \\
x'_3 &= v_1 x_1 + v_2 x_2 + x_3
\end{aligned}
$$

Given $x = \frac{x_1}{x_3}$ and $y = \frac{x_2}{x_3}$, the above two equations can be further simplified:

$$
\begin{aligned}
x' &= \frac{a_{11}x + a_{12}y + t_x}{v_1 x + v_2 y + 1} \\
y' &= \frac{a_{21}x + a_{22}y + t_y}{v_1 x + v_2 y + 1}
\end{aligned}
$$

$$
\begin{aligned}
a_{11}x + a_{12}y + t_x - v_1 x x' - v_2 y x' &= x' \\
a_{21}x + a_{22}y + t_y - v_1 x y' - v_2 y y' &= y'
\end{aligned}
$$

Hence there are 8 unknowns in $\mathbf{H}$, we need at least 4 correspondence points to solve them. After plugging 4 correspondence points, we get 8 equations:

$$
\begin{aligned}
a_{11}x_1 + a_{12}y_1 + t_x + 0a_{21} + 0a_{22} + 0t_y - v_1 x_1 x'_1 - v_2 y_1 x'_1 &= x'_1 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_1 + a_{22}y_1 + t_y - v_1 x_1 y'_1 - v_2 y_1 y'_1 &= y'_1 \\
a_{11}x_2 + a_{12}y_2 + t_x + 0a_{21} + 0a_{22} + 0t_y - v_1 x_2 x'_2 - v_2 y_2 x'_2 &= x'_2 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_2 + a_{22}y_2 + t_y - v_1 x_2 y'_2 - v_2 y_2 y'_2 &= y'_2 \\
a_{11}x_3 + a_{12}y_3 + t_x + 0a_{21} + 0a_{22} + 0t_y - v_1 x_3 x'_3 - v_2 y_3 x'_3 &= x'_3 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_3 + a_{22}y_3 + t_y - v_1 x_3 y'_3 - v_2 y_3 y'_3 &= y'_3 \\
a_{11}x_4 + a_{12}y_4 + t_x + 0a_{21} + 0a_{22} + 0t_y - v_1 x_4 x'_4 - v_2 y_4 x'_4 &= x'_4 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_4 + a_{22}y_4 + t_y - v_1 x_4 y'_4 - v_2 y_4 y'_4 &= y'_4
\end{aligned}
$$

By solving these linear equations, we can get entries of $\mathbf{H}$.

Logic - Solve an overdetermined system

When there are more correspondence points than the number of parameters, least squares method is used to find the solution [1].

Logic - Inverse Image Warping

When projecting the ROI in the domain space into the frame PQRS in the range space, cooridates of the transformed point (x', y') may not be integers, and if we simply turn them into integers using the floor or ceil function, there will be "holes" (pixels that don't get mapped to) in the output image. Therefore, in order to resolve this issue, we use inverse image warping, that is, for each point in the PQRS frame, we find the input point that maps to it in the domain space by using

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{H^{-1}} \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$$

and update the pixel value in three color channels at (x', y') to pixel values at (x, y).

If coordinates of the input point (x, y) are not integers, then we estimate pixel value p^* in every color channel at (x, y) using Bilinear Interpolation, which calculates the pixel value using the distance-weighted value of the four nearest pixels [2].

$$p^* = \frac{\sum_{i=1}^{4} \frac{1}{d_i} p_i}{\sum_{i=1}^{4} \frac{1}{d_i}}$$

where d_i is the distance between the input point and its i^{th} neighbour and p_i is the pixel value of a specific color channel of the i^{th} neighbour.

Task 1.1

Steps

Step 1: Find coordinates of points P, Q, R and S and find 4 points from the car image.

Step 2: Compute the $\mathbf{H}$ matrix as described above.

Step 3: Extract the PQRS frame region in the range space.

Step 4: For every point (x', y') ($\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$ in $\mathbb{R}^3$, where $x' = \frac{x'_1}{x'_3}$ and $y' = \frac{x'_2}{x'_3}$) in the PQRS region, compute its correspondence point (x, y) ($\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ in $\mathbb{R}^3$, where $x = \frac{x_1}{x_3}$ and $y = \frac{x_2}{x_3}$) in the domain space using

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{H}^{-1} \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$$

and the pixel value in each color channel at (x', y') to the pixel value at (x, y) (Inverse Image Warping). If x or y is not a integer, using bilinear interpolation as described above to get an estimated pixel value at (x, y).

Functions

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
import matplotlib.image as img
import numpy as np
import cv2
import math
```

In [3]:
```
def findHomographyMatrix(a, b):
    ans = np.matmul(np.linalg.inv(a), np.transpose(b))
    ans = np.append(ans, [1])
    H = ans.reshape(3, 3)
    return H
```

In [4]:
```
def findHomographyMatrixLeastSquares(a, b):
    aTranspose = a.T
    a = np.matmul(aTranspose, a)
    b = np.matmul(aTranspose, b)
    ans = np.linalg.solve(a, b)
    ans = np.append(ans, [0, 0, 1])
    H = ans.reshape(3, 3)
    return H
```

In [5]:
```
def extractPQRS(w, h, cornerTL, cornerBL, cornerBR, cornerTR):
    newImg = np.zeros((w, h),dtype='uint8')
    points = np.array([cornerTL, cornerBL, cornerBR, cornerTR])
    cv2.fillPoly(newImg, pts=[points], color=(255, 0, 0))
    return newImg
```

In [6]:
```
def performBilinearInterpolation(domainImg, rangeImg, rangeX, rangeY, domair
    neighbour1 = [np.floor(domainX).astype(int), np.floor(domainY).astype(ir
    neighbour2 = [np.ceil(domainX).astype(int), np.floor(domainY).astype(int
    neighbour3 = [np.floor(domainX).astype(int), np.ceil(domainY).astype(int
    neighbour4 = [np.ceil(domainX).astype(int), np.ceil(domainY).astype(int)

    d1 = math.sqrt((neighbour1[0] - domainX)**2 + (neighbour1[1] - domainY)*
    d2 = math.sqrt((neighbour2[0] - domainX)**2 + (neighbour2[1] - domainY)*
    d3 = math.sqrt((neighbour3[0] - domainX)**2 + (neighbour3[1] - domainY)*
    d4 = math.sqrt((neighbour4[0] - domainX)**2 + (neighbour4[1] - domainY)*

    for colorChannel in range(3):
        p1 = domainImg[neighbour1[1], neighbour1[0], colorChannel]
        p2 = domainImg[neighbour2[1], neighbour2[0], colorChannel]
        p3 = domainImg[neighbour3[1], neighbour3[0], colorChannel]
        p4 = domainImg[neighbour4[1], neighbour4[0], colorChannel]

        estimatedV = (p1/d1 + p2/d2 + p3/d3 + p4/d4) / (1/d1 + 1/d2 + 1/d3 +
        rangeImg[rangeY, rangeX, colorChannel] = estimatedV

    return rangeImg
```

In [7]:
```
def inverseImgWarping(domainImg, rangeImg, ROI, H):
    for i in range(rangeImg.shape[1]):
        for j in range(rangeImg.shape[0]):
            if(ROI[j, i] == 255):
                [newX, newY, third] = np.dot(np.linalg.pinv(H), np.transpose
                intY = np.floor(newY/third).astype(int)
                intX = np.floor(newX/third).astype(int)
                if(newY/third > 0 and newY/third > 0 and intY < domainImg.sh
                    if(intX == newX and intY == newY):
                        rangeImg[j, i] = domainImg[intY, intX]
                    else:
                        rangeImg = performBilinearInterpolation(domainImg, r
    return rangeImg
```

In [8]:
```
domainImg = img.imread('./hw2images/car.jpg')
card1 = img.imread('./hw2images/card1.jpeg')
card2 = img.imread('./hw2images/card2.jpeg')
card3 = img.imread('./hw2images/card3.jpeg')
```

Image Card 1

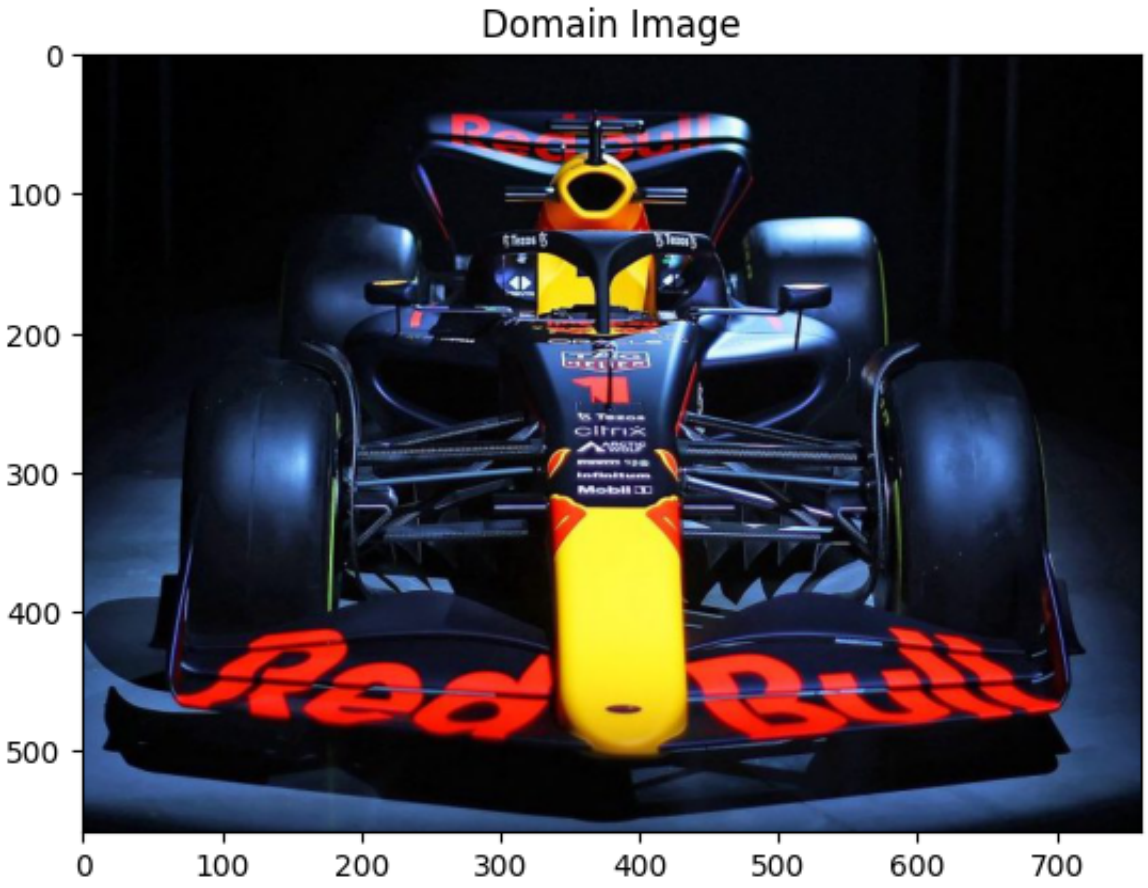
4 correspondence points

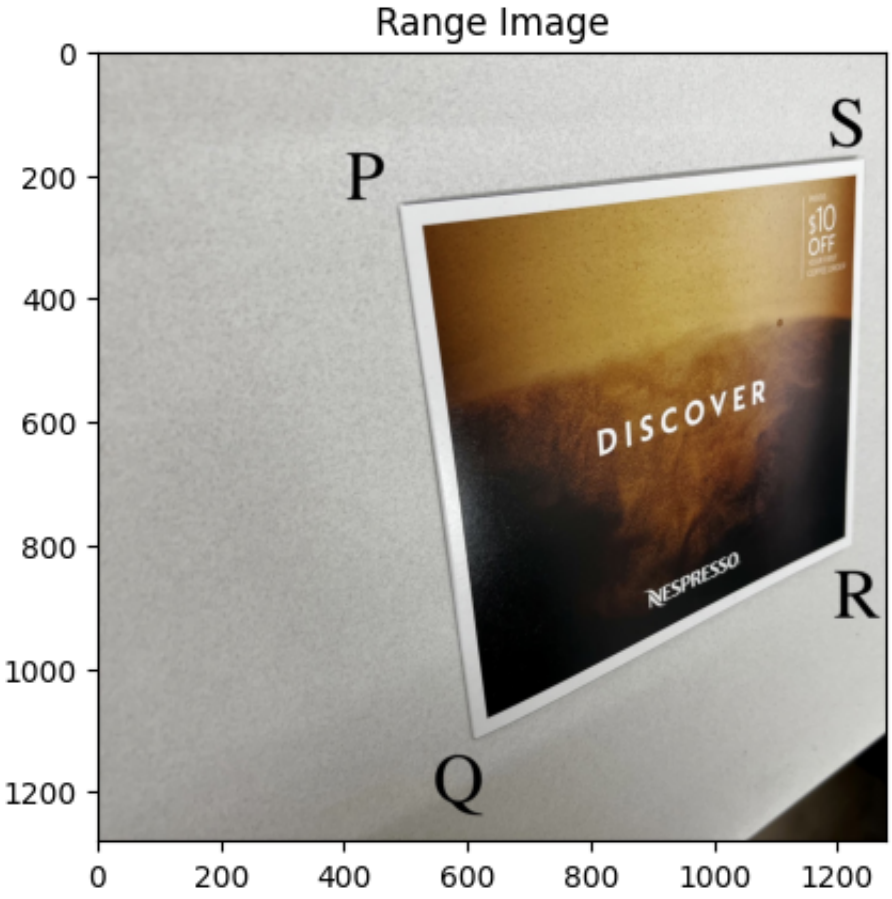
$$\begin{aligned} \text{Domain Space} &\rightarrow \text{Range Space} \\ A(40, 34) &\rightarrow P(491, 254) \\ B(40, 534) &\rightarrow Q(610, 1121) \\ C(711, 34) &\rightarrow S(1240, 179) \\ D(711, 534) &\rightarrow R(1221, 806) \end{aligned}$$

The 8 equations are:

$$\begin{aligned} 40a_{11} + 34a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 19640v_1 - 16694v_2 &= 491 \\ 0a_{11} + 0a_{12} + 0t_x + 40a_{21} + 34a_{22} + t_y - 10160v_1 - 8636v_2 &= 254 \\ 40a_{11} + 534a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 24400v_1 - 325740v_2 &= 610 \\ 0a_{11} + 0a_{12} + 0t_x + 40a_{21} + 534a_{22} + t_y - 44840v_1 - 598614v_2 &= 1121 \\ 711a_{11} + 34a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 881640v_1 - 42160v_2 &= 1240 \\ 0a_{11} + 0a_{12} + 0t_x + 711a_{21} + 34a_{22} + t_y - 127269v_1 - 6086v_2 &= 179 \\ 711a_{11} + 534a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 868131v_1 - 652014v_2 &= 1221 \\ 0a_{11} + 0a_{12} + 0t_x + 711a_{21} + 534a_{22} + t_y - 573066v_1 - 430404v_2 &= 806 \end{aligned}$$

In [9]:
```
a = np.array([[40, 34, 1, 0, 0, 0, -19640, -16694],
              [0, 0, 0, 40, 34, 1, -10160, -8638],
              [40, 534, 1, 0, 0, 0, -24400, -325740],
              [0, 0, 0, 40, 534, 1, -44840, -598614],
              [711, 34, 1, 0, 0, 0, -881640, -42160],
              [0, 0, 0, 711, 34, 1, -127269, -6086],
              [711, 534, 1, 0, 0, 0, -868131, -652014],
              [0, 0, 0, 711, 534, 1, -573066, -430404]])

b = np.array([491, 254, 610, 1121, 1240, 179, 1221, 806])

H = findHomographyMatrix(a, b)

ROIImg = extractPQRS(card1.shape[0], card1.shape[1], [491, 254], [610, 1121]

newImg = inverseImgWarping(domainImg, np.copy(card1), ROIImg, H)

plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(card1)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task1.1_carToCard1.png')
```


Domain Image
0
100
200
300
400
500
0
100
200
300
400
500
600
700


Range Image
P
S
Q
R
DISCOVER
NESPRESSO
$10
OFF
0
200
400
600
800
1000
1200

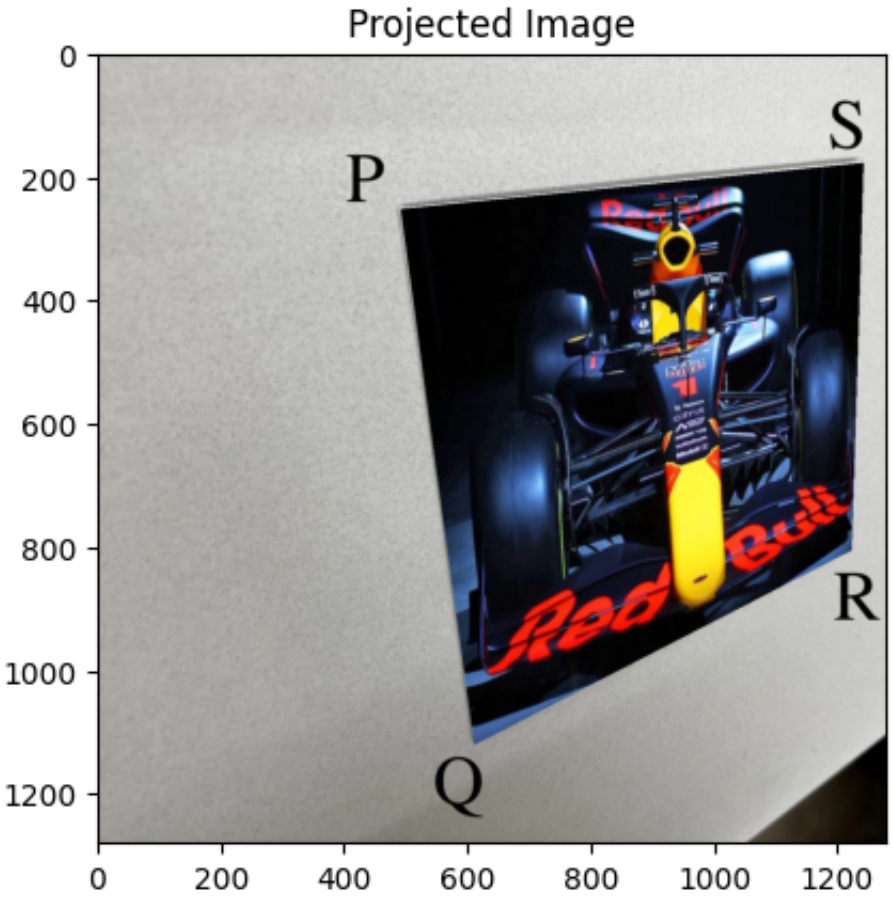

Projected Image
P
S
R
Q
0
200
400
600
800
1000
1200

Image Card 2

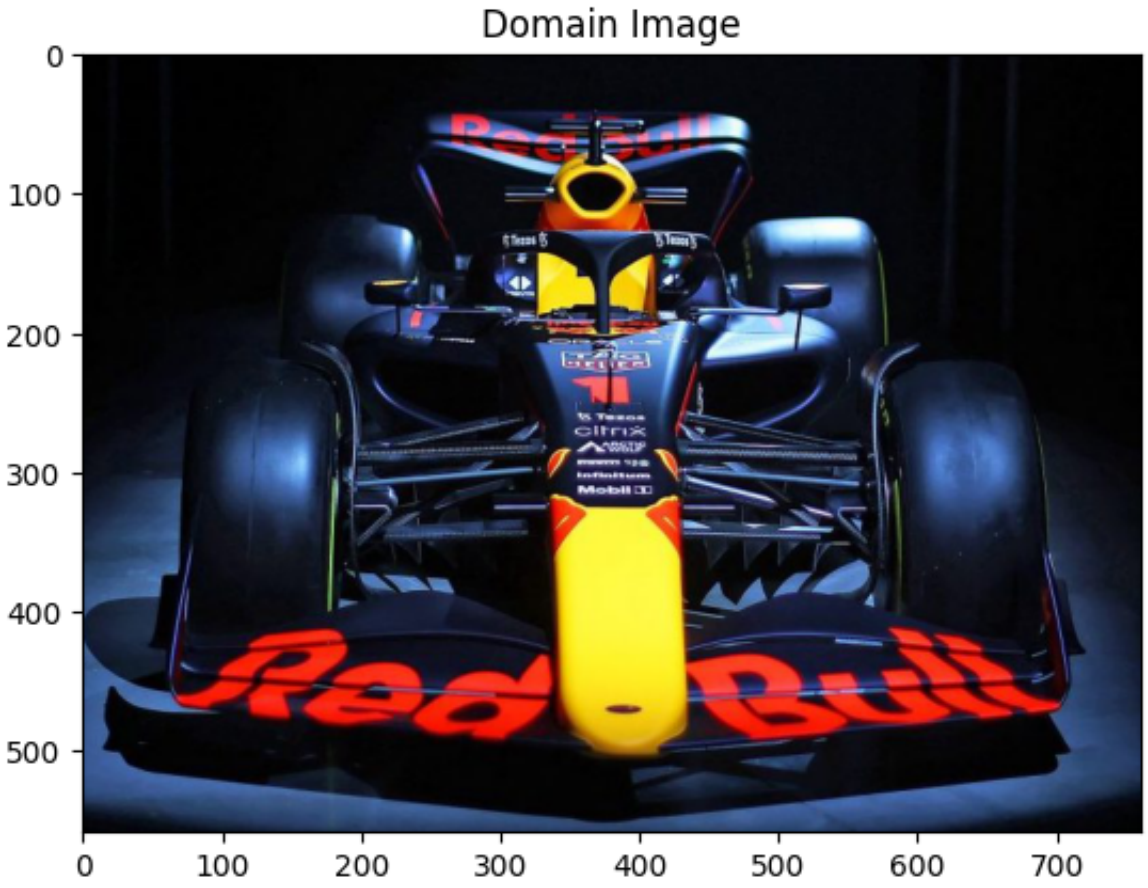
4 correspondence points

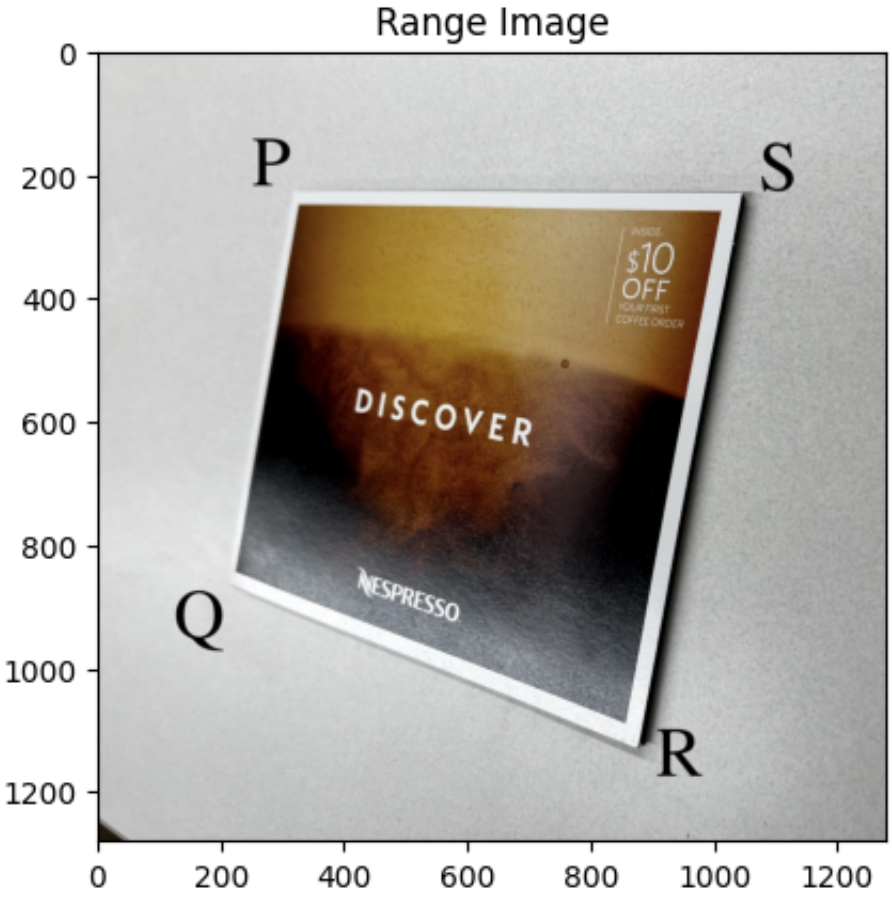
$$\begin{aligned}\text{Domain Space} &\rightarrow \text{Range Space}\\ A(40,34) &\rightarrow P(320,232)\\ B(40,534) &\rightarrow Q(210,863)\\ C(711,34) &\rightarrow S(1045,235)\\ D(711,534) &\rightarrow R(875,1133)\end{aligned}$$

The 8 equations are:

$$\begin{aligned}40a_{11}+34a_{12}+t_x+0a_{21}+0a_{22}+0t_y-12800v_1-10880v_2&=320\\ 0a_{11}+0a_{12}+0t_x+40a_{21}+34a_{22}+t_y-9280v_1-7888v_2&=232\\ 40a_{11}+534a_{12}+t_x+0a_{21}+0a_{22}+0t_y-8400v_1-112140v_2&=210\\ 0a_{11}+0a_{12}+0t_x+40a_{21}+534a_{22}+t_y-34520v_1-460842v_2&=863\\ 711a_{11}+34a_{12}+t_x+0a_{21}+0a_{22}+0t_y-742995v_1-35530v_2&=1045\\ 0a_{11}+0a_{12}+0t_x+711a_{21}+34a_{22}+t_y-167085v_1-7990v_2&=235\\ 711a_{11}+534a_{12}+t_x+0a_{21}+0a_{22}+0t_y-622125v_1-467250v_2&=875\\ 0a_{11}+0a_{12}+0t_x+711a_{21}+534a_{22}+t_y-805563v_1-605022v_2&=1133\end{aligned}$$

In [63]:
```
a = np.array([[40, 34, 1, 0, 0, 0, -12800, -10880],
              [0, 0, 0, 40, 34, 1, -9280, -7888],
              [40, 534, 1, 0, 0, 0, -8400, -112140],
              [0, 0, 0, 40, 534, 1, -34520, -460842],
              [711, 34, 1, 0, 0, 0, -742995, -35530],
              [0, 0, 0, 711, 34, 1, -167085, -7990],
              [711, 534, 1, 0, 0, 0, -622125, -467250],
              [0, 0, 0, 711, 534, 1, -805563, -605022]])

b = np.array([320, 232, 210, 863, 1045, 235, 875, 1133])

H = findHomographyMatrix(a, b)

ROIImg = extractPQRS(card2.shape[0], card2.shape[1], [320, 232], [210, 863],

newImg = inverseImgWarping(domainImg, np.copy(card2), ROIImg, H)

plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(card2)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task1.1_carToCard2.png')
```


Domain Image
0
100
200
300
400
500
0
100
200
300
400
500
600
700


Range Image
P
S
Q
R
DISCOVER
NESPRESSO
$10
OFF
0
200
400
600
800
1000
1200

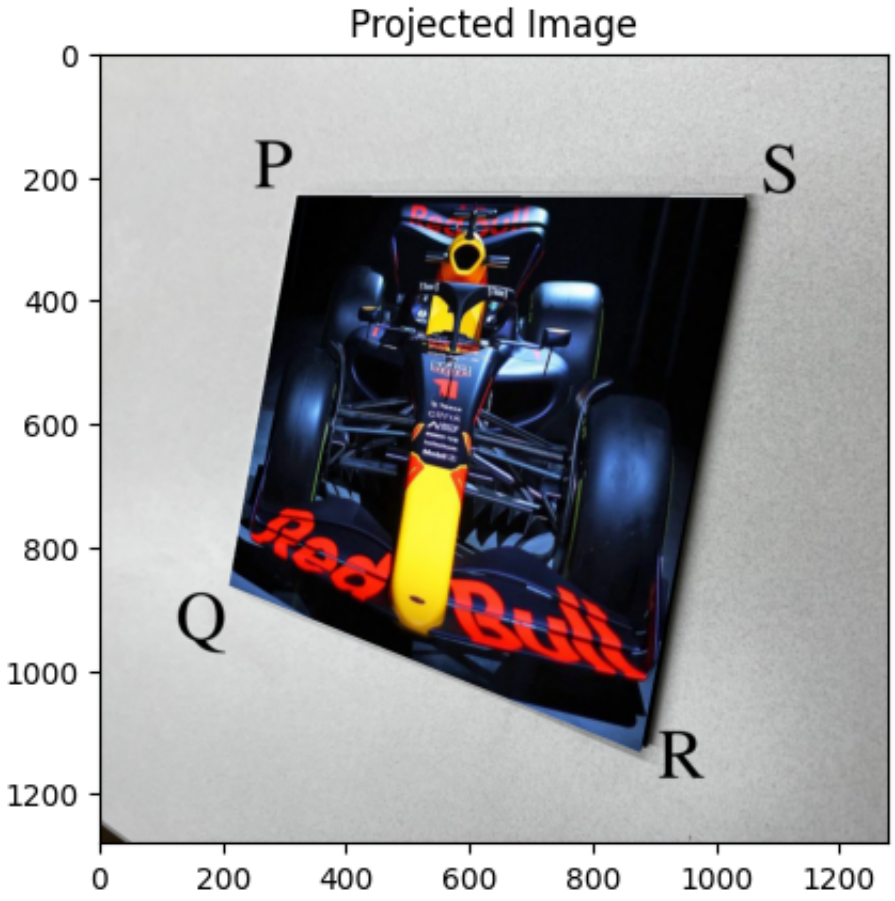


Image Card 3

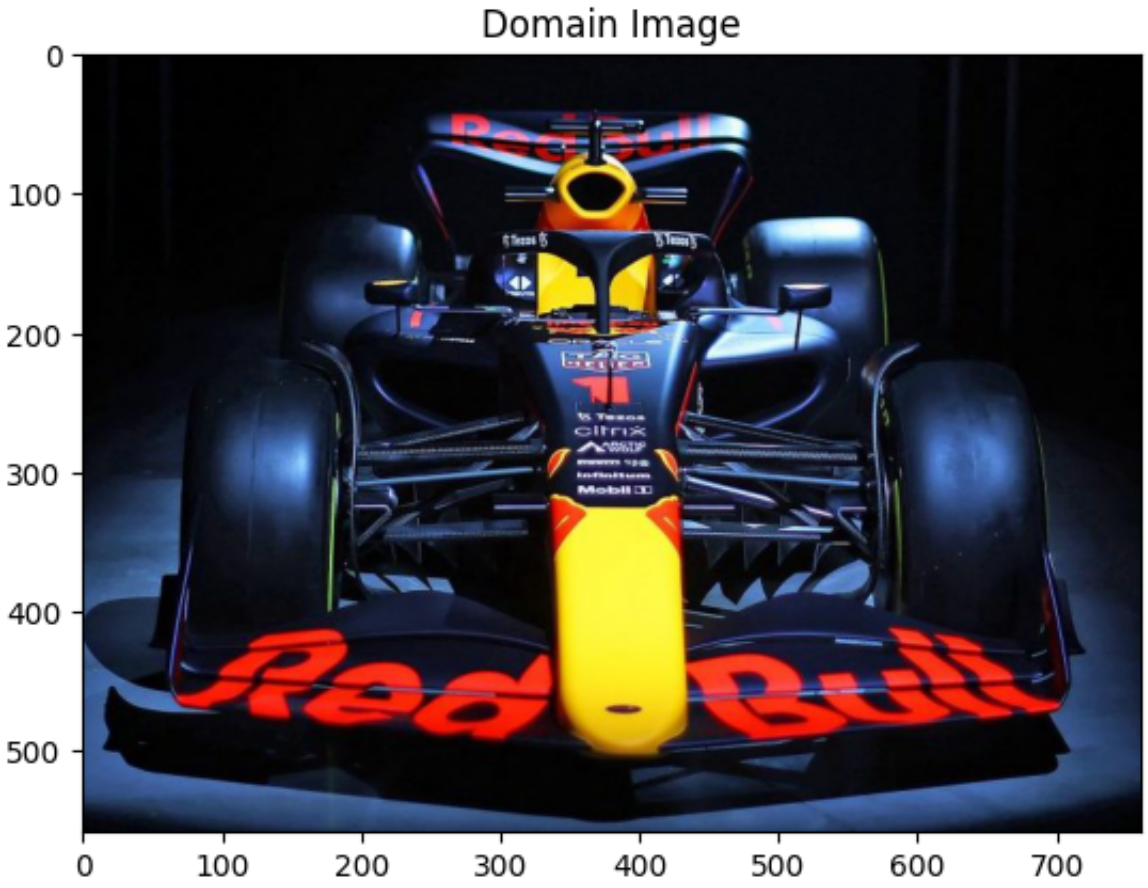
4 correspondence points

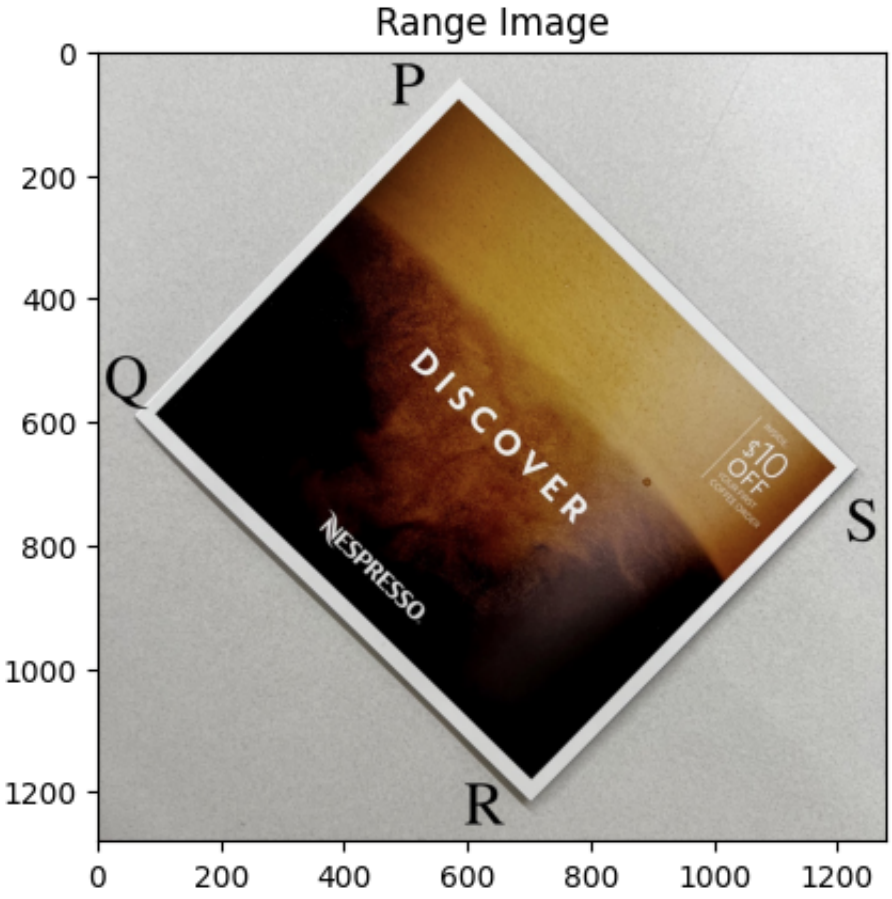
$$\begin{aligned}\text{Domain Space} &\rightarrow \text{Range Space}\\ A(40,34) &\rightarrow P(587,48)\\ B(40,534) &\rightarrow Q(62,594)\\ C(711,34) &\rightarrow S(1232,678)\\ D(711,534) &\rightarrow R(702,1219)\end{aligned}$$

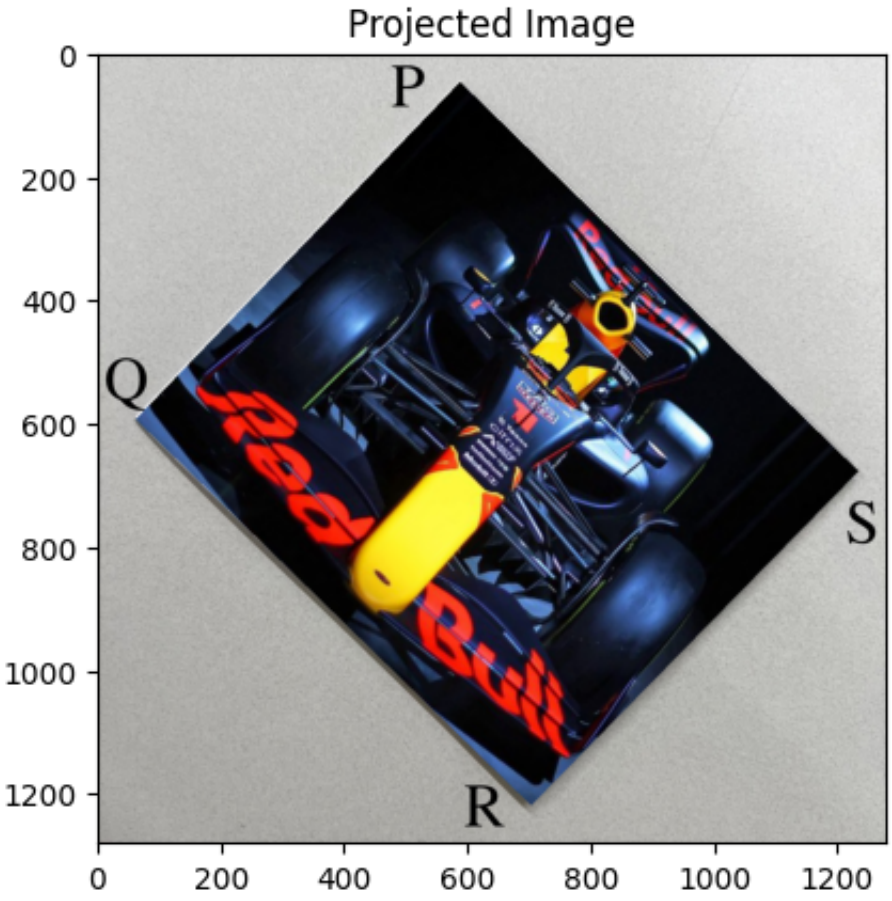
The 8 equations are:

$$\begin{aligned}40a_{11}+34a_{12}+t_x+0a_{21}+0a_{22}+0t_y-23480v_1-19958v_2&=587\\ 0a_{11}+0a_{12}+0t_x+40a_{21}+34a_{22}+t_y-1920v_1-1632v_2&=48\\ 40a_{11}+534a_{12}+t_x+0a_{21}+0a_{22}+0t_y-2480v_1-33108v_2&=62\\ 0a_{11}+0a_{12}+0t_x+40a_{21}+534a_{22}+t_y-23760v_1-317196v_2&=594\\ 711a_{11}+34a_{12}+t_x+0a_{21}+0a_{22}+0t_y-875952v_1-41888v_2&=1232\\ 0a_{11}+0a_{12}+0t_x+711a_{21}+34a_{22}+t_y-482058v_1-23052v_2&=678\\ 711a_{11}+534a_{12}+t_x+0a_{21}+0a_{22}+0t_y-499122v_1-374868v_2&=702\\ 0a_{11}+0a_{12}+0t_x+711a_{21}+534a_{22}+t_y-866709v_1-650946v_2&=1219\end{aligned}$$

In [65]:
```
a = np.array([[40, 34, 1, 0, 0, 0, -23480, -19958],
              [0, 0, 0, 40, 34, 1, -1920, -1632],
              [40, 534, 1, 0, 0, 0, -2480, -33108],
              [0, 0, 0, 40, 534, 1, -23760, -317196],
              [711, 34, 1, 0, 0, 0, -875952, -41888],
              [0, 0, 0, 711, 34, 1, -482058, -23052],
              [711, 534, 1, 0, 0, 0, -499122, -374868],
              [0, 0, 0, 711, 534, 1, -866709, -650946]])
b = np.array([587, 48, 62, 594, 1232, 678, 702, 1219])

H = findHomographyMatrix(a, b)

ROIImg = extractPQRS(card3.shape[0], card3.shape[1], [587, 48], [62, 594], [

newImg = inverseImgWarping(domainImg, np.copy(card3), ROIImg, H)

plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(card3)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task1.1_carToCard3.png')
```


Domain Image
0
100
200
300
400
500
0
100
200
300
400
500
600
700


Range Image
P
Q
S
R
DISCOVER
NESPRESSO
$10
OFF
0
200
400
600
800
1000
1200


Projected Image
P
Q
S
R
0
200
400
600
800
1000
1200

Task 1.2

Step 1: Find the homography matrix $\mathbf{H_{ab}}$ using the logic described above and the correspondence points in this case are corners (P, Q, R and S) in Figs. 1a and 1b. Similarly, we can find the homography matrix $\mathbf{H_{bc}}$ between Figs. 1b and 1c.

Step 2: Compute the product of $\mathbf{H_{bc}}$ and $\mathbf{H_{ab}}$.

Step 3: Extract ROI in Fig. 1c, which is the entire image.

Step 4: For each point in ROI (Fig. 1c), use inverse image wraping to find its correspondence point in Fig. 1a. If coordinates of the calculated point are not integers, using bilinear interpolation as described above to get an estimated pixel value.

Correspondence points in Figs. 1a and 1b

$$\begin{gathered}\text{Fig. 1a} \rightarrow \text{Fig. 1b} \\ (491, 254) \rightarrow (320, 232) \\ (1240, 179) \rightarrow (1045, 235) \\ (610, 1121) \rightarrow (210, 863) \\ (1221, 806) \rightarrow (875, 1133)\end{gathered}$$

The 8 equations are:

$$\begin{gathered}491a_{11} + 254a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 157120v_1 - 81280v_2 = 320 \\ 0a_{11} + 0a_{12} + 0t_x + 491a_{21} + 254a_{22} + t_y - 113912v_1 - 58928v_2 = 232 \\ 1240a_{11} + 179a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 1295800v_1 - 187055v_2 = 1045 \\ 0a_{11} + 0a_{12} + 0t_x + 1240a_{21} + 179a_{22} + t_y - 291400v_1 - 42065v_2 = 235 \\ 610a_{11} + 1121a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 128100v_1 - 235410v_2 = 210 \\ 0a_{11} + 0a_{12} + 0t_x + 610a_{21} + 1121a_{22} + t_y - 526430v_1 - 967423v_2 = 863 \\ 1221a_{11} + 806a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 1068375v_1 - 705250v_2 = 875 \\ 0a_{11} + 0a_{12} + 0t_x + 1221a_{21} + 806a_{22} + t_y - 1383393v_1 - 913198v_2 = 1133\end{gathered}$$

In [66]:
```
a = np.array([[491, 254, 1, 0, 0, 0, -157120, -81280],
              [0, 0, 0, 491, 254, 1, -113912, -58928],
              [1240, 179, 1, 0, 0, 0, -1295800, -187055],
              [0, 0, 0, 1240, 179, 1, -291400, -42065],
              [610, 1121, 1, 0, 0, 0, -128100, -235410],
              [0, 0, 0, 610, 1121, 1, -526430, -967423],
              [1221, 806, 1, 0, 0, 0, -1068375, -705250],
              [0, 0, 0, 1221, 806, 1, -1383393, -913198],
             ])

b = np.array([320, 232, 1045, 235, 210, 863, 875, 1133])

H1 = findHomographyMatrix(a, b)
```

Correspondence points in Figs. 1b and 1c

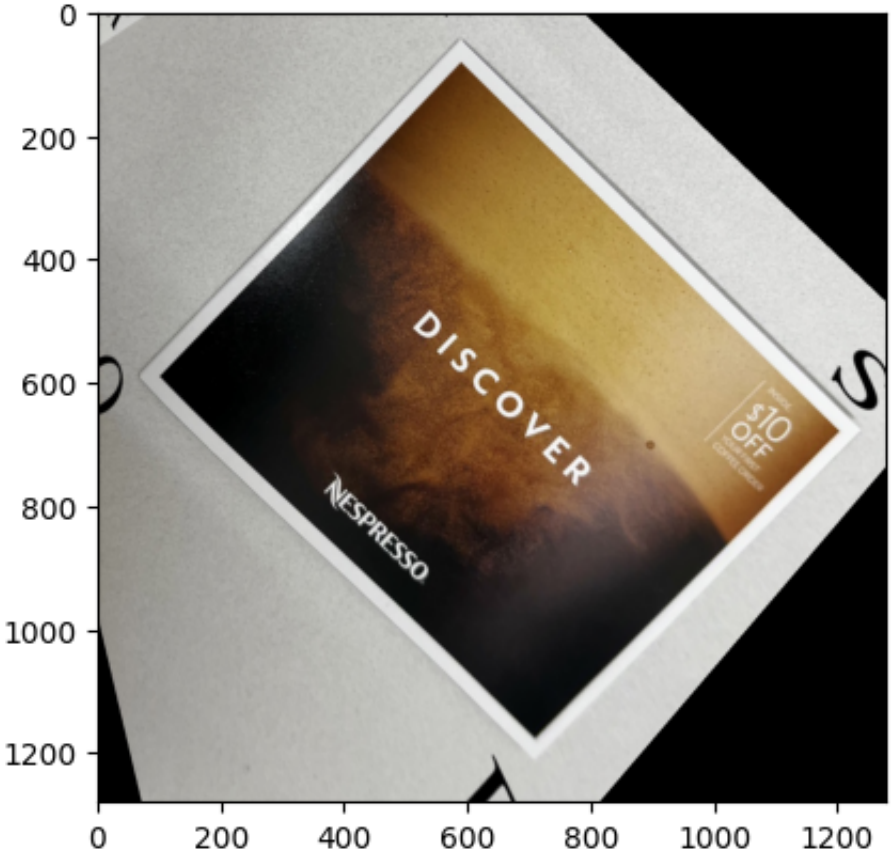
$$\begin{gathered}\text{Fig. 1b} \rightarrow \text{Fig. 1c} \\ (320, 232) \rightarrow (587, 48) \\ (1045, 235) \rightarrow (1232, 678) \\ (210, 863) \rightarrow (62, 594) \\ (875, 1133) \rightarrow (702, 1219)\end{gathered}$$

The 8 equations are:

$$\begin{gathered}320a_{11} + 232a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 187840v_1 - 136184v_2 = 587 \\ 0a_{11} + 0a_{12} + 0t_x + 320a_{21} + 232a_{22} + t_y - 15360v_1 - 11136v_2 = 48 \\ 1045a_{11} + 235a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 1287440v_1 - 289520v_2 = 1232 \\ 0a_{11} + 0a_{12} + 0t_x + 1045a_{21} + 235a_{22} + t_y - 708510v_1 - 159330v_2 = 678 \\ 210a_{11} + 863a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 13020v_1 - 53506v_2 = 62 \\ 0a_{11} + 0a_{12} + 0t_x + 210a_{21} + 863a_{22} + t_y - 124740v_1 - 512622v_2 = 594 \\ 875a_{11} + 1133a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 614250v_1 - 795366v_2 = 702 \\ 0a_{11} + 0a_{12} + 0t_x + 875a_{21} + 1133a_{22} + t_y - 1066625v_1 - 1381127v_2 = 1219\end{gathered}$$

In [67]:
```
a = np.array([[320, 232, 1, 0, 0, 0, -187840, -136184],
              [0, 0, 0, 320, 232, 1, -15360, -11136],
              [1045, 235, 1, 0, 0, 0, -1287440, -289520],
              [0, 0, 0, 1045, 235, 1, -708510, -159330],
              [210, 863, 1, 0, 0, 0, -13020, -53506],
              [0, 0, 0, 210, 863, 1, -124740, -512622],
              [875, 1133, 1, 0, 0, 0, -614250, -795366],
              [0, 0, 0, 875, 1133, 1, -1066625, -1381127],
             ])

b = np.array([587, 48, 1232, 678, 62, 594, 702, 1219])

H2 = findHomographyMatrix(a, b)
```

In [68]:
```
# Find the product of H2 and H1
H = np.matmul(H2, H1)


ROIImg = extractPQRS(card3.shape[0], card3.shape[1], [0, 0], [0, 1279], [127


ans = np.zeros((card1.shape[0], card1.shape[1], 3), dtype='uint8')


newImg = inverseImgWarping(card1, ans, ROIImg, H)


fig = plt.figure()
plt.imshow(newImg)
plt.savefig('./hw2images/task1.2_card1ToCard3.png')
```


0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
DISCOVER
NESPRESSO
$10
OFF

Task 1.3

A homography is affine if its last row is [0, 0, 1], and thus, we have

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

When using 4 correspondence points, we get the following 8 equations:

$$\begin{aligned}
a_{11}x_1 + a_{12}y_1 + t_x + 0a_{21} + 0a_{22} + 0t_y &= x'_1 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_1 + a_{22}y_1 + t_y &= y'_1 \\
a_{11}x_2 + a_{12}y_2 + t_x + 0a_{21} + 0a_{22} + 0t_y &= x'_2 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_2 + a_{22}y_2 + t_y &= y'_2 \\
a_{11}x_3 + a_{12}y_3 + t_x + 0a_{21} + 0a_{22} + 0t_y &= x'_3 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_3 + a_{22}y_3 + t_y &= y'_3 \\
a_{11}x_4 + a_{12}y_4 + t_x + 0a_{21} + 0a_{22} + 0t_y &= x'_4 \\
0a_{11} + 0a_{12} + 0t_x + a_{21}x_4 + a_{22}y_4 + t_y &= y'_4
\end{aligned}$$

Steps

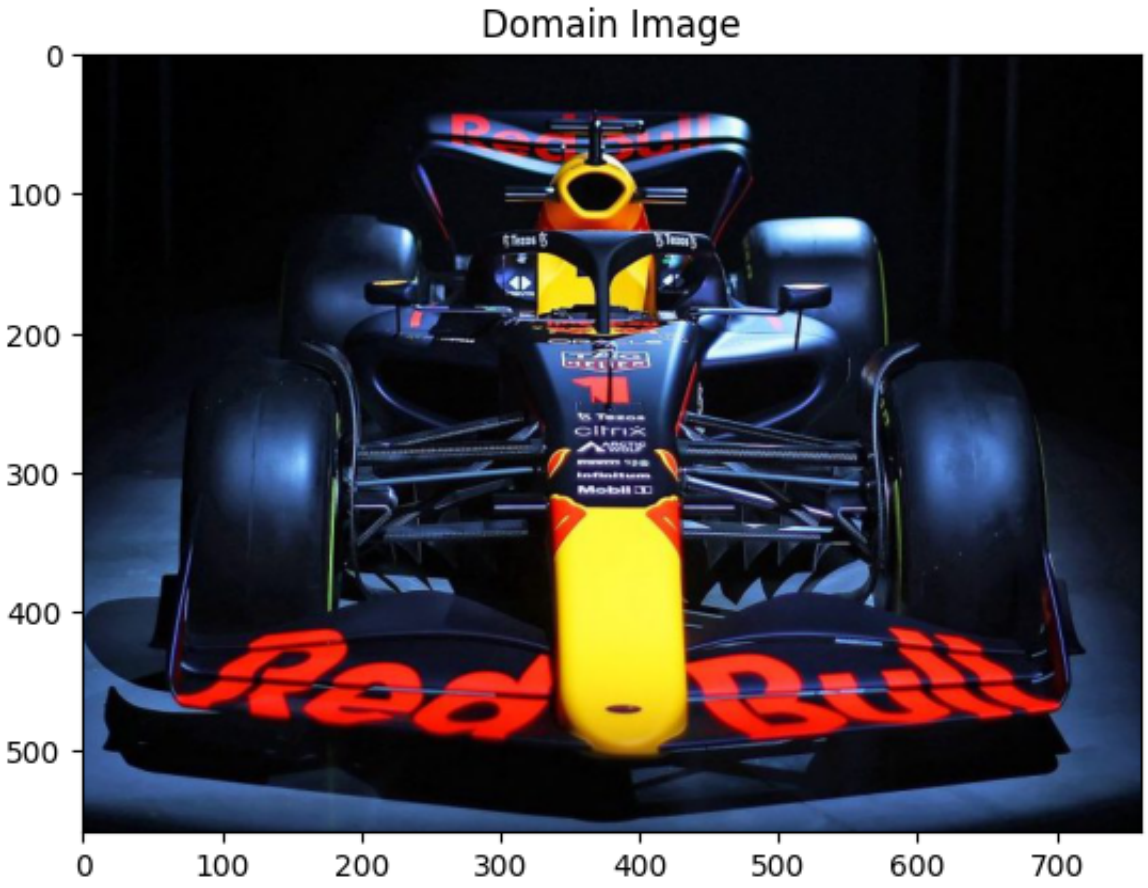
Step 1: Find correspondence points and compute the homography matrix H using the logic described above. However, since we now have an overdetermined system (8 linear equations and 6 unknowns), we need to use least square method to solve the overdetermined system.

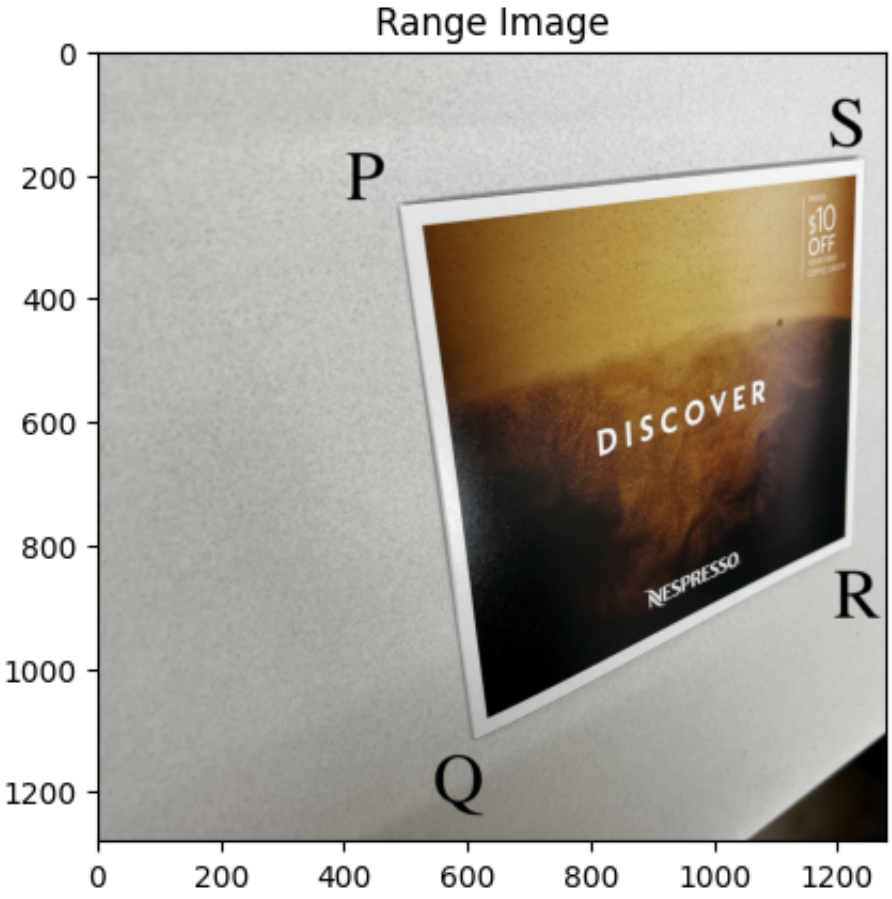
Step 2: Extract the PQRS frame region in the range space.

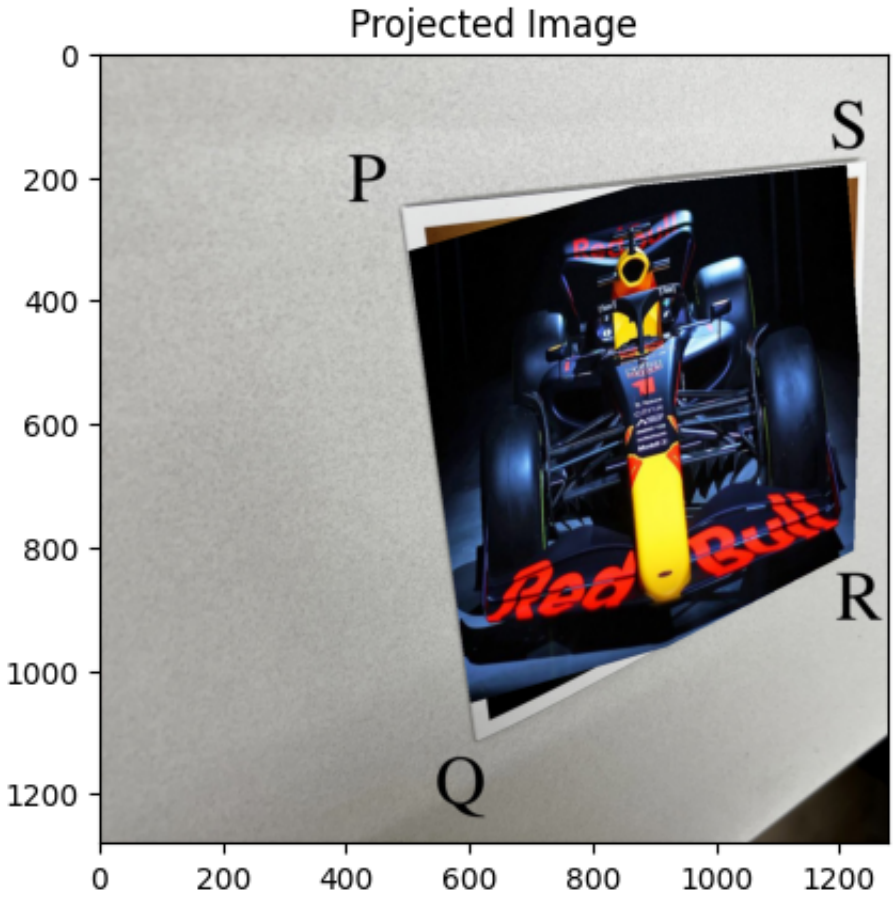
Step 3: Perform inverse image warping to project the entire car image to the PQRS frame region. Use bilinear interpolation to get an estimated pixel value if the caculated coordinates are not integers.

Card 1

In [69]:
```
a = np.array([[0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 1],
              [0, 558, 1, 0, 0, 0],
              [0, 0, 0, 0, 558, 1],
              [760, 0, 1, 0, 0, 0],
              [0, 0, 0, 760, 0, 1],
              [760, 558, 1, 0, 0, 0],
              [0, 0, 0, 760, 558, 1]
             ])

b = np.array([491, 254, 610, 1121, 1240, 179, 1221, 806])

H = findHomographyMatrixLeastSquares(a, b)


ROIImg = extractPQRS(card1.shape[0], card1.shape[1], [491, 254], [610, 1121]


newImg = inverseImgWarping(domainImg, np.copy(card1), ROIImg, H)


plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(card1)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task1.3_carToCard1.png')
```

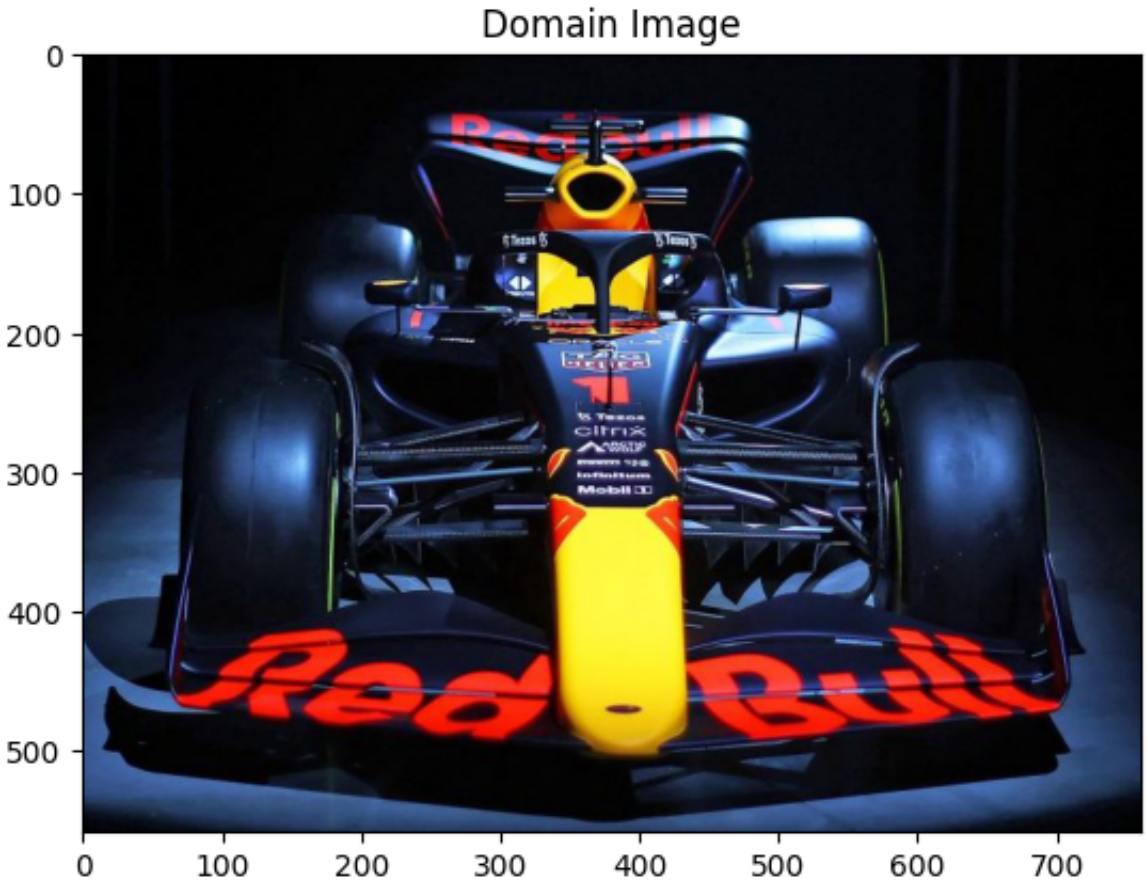

Domain Image
0
100
200
300
400
500
0
100
200
300
400
500
600
700

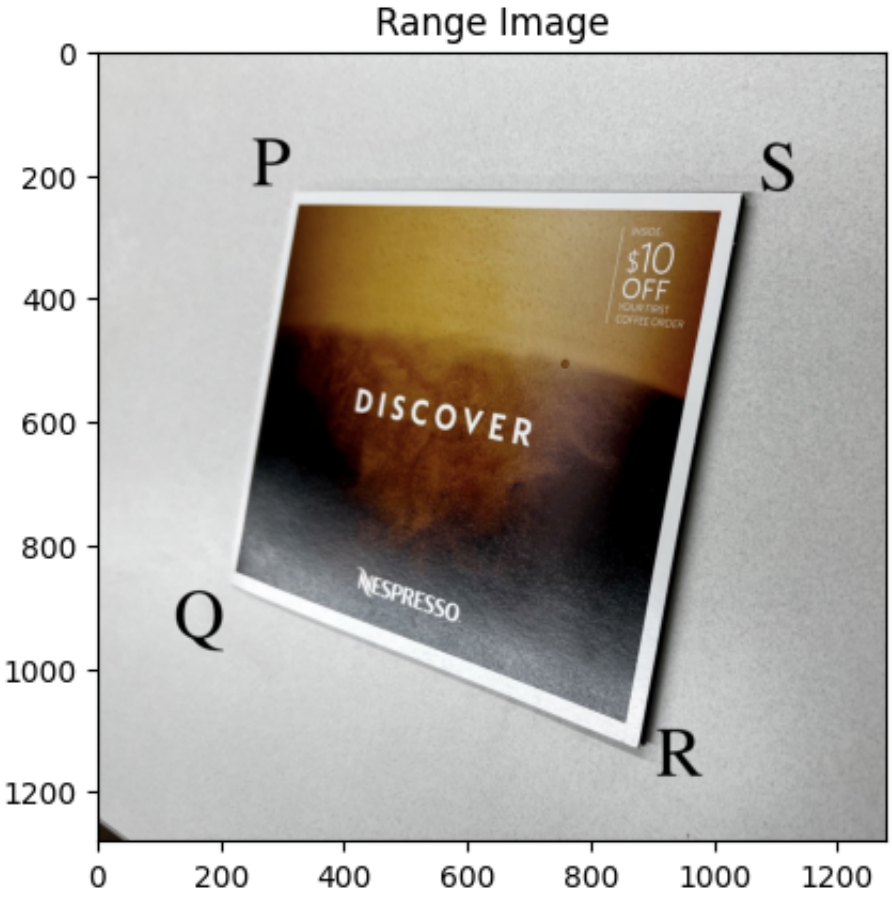

Range Image
P
S
Q
R
DISCOVER
NESPRESSO
$10
OFF
0
200
400
600
800
1000
1200

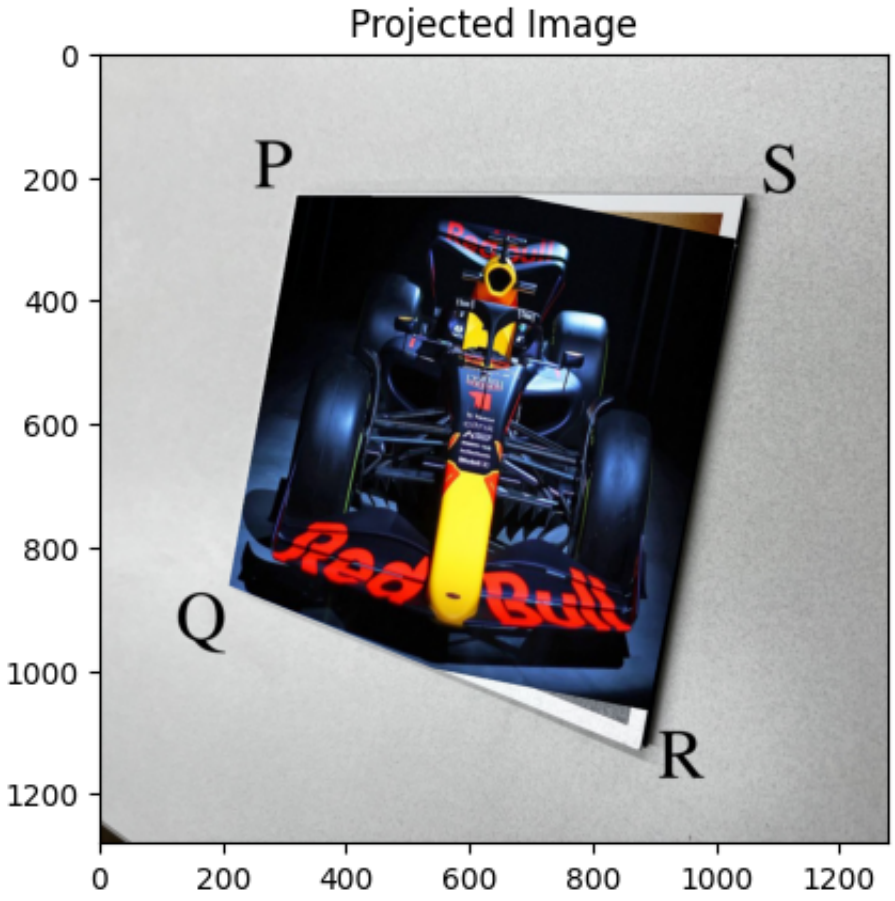


Card 2

In [70]:
```
a = np.array([[0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 1],
              [0, 558, 1, 0, 0, 0],
              [0, 0, 0, 0, 558, 1],
              [760, 0, 1, 0, 0, 0],
              [0, 0, 0, 760, 0, 1],
              [760, 558, 1, 0, 0, 0],
              [0, 0, 0, 760, 558, 1]])

b = np.array([320, 232, 210, 863, 1045, 235, 875, 1133])

H = findHomographyMatrixLeastSquares(a, b)

ROIImg = extractPQRS(card2.shape[0], card2.shape[1], [320, 232], [210, 863],


newImg = inverseImgWarping(domainImg, np.copy(card2), ROIImg, H)


plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(card2)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task1.3_carToCard2.png')
```

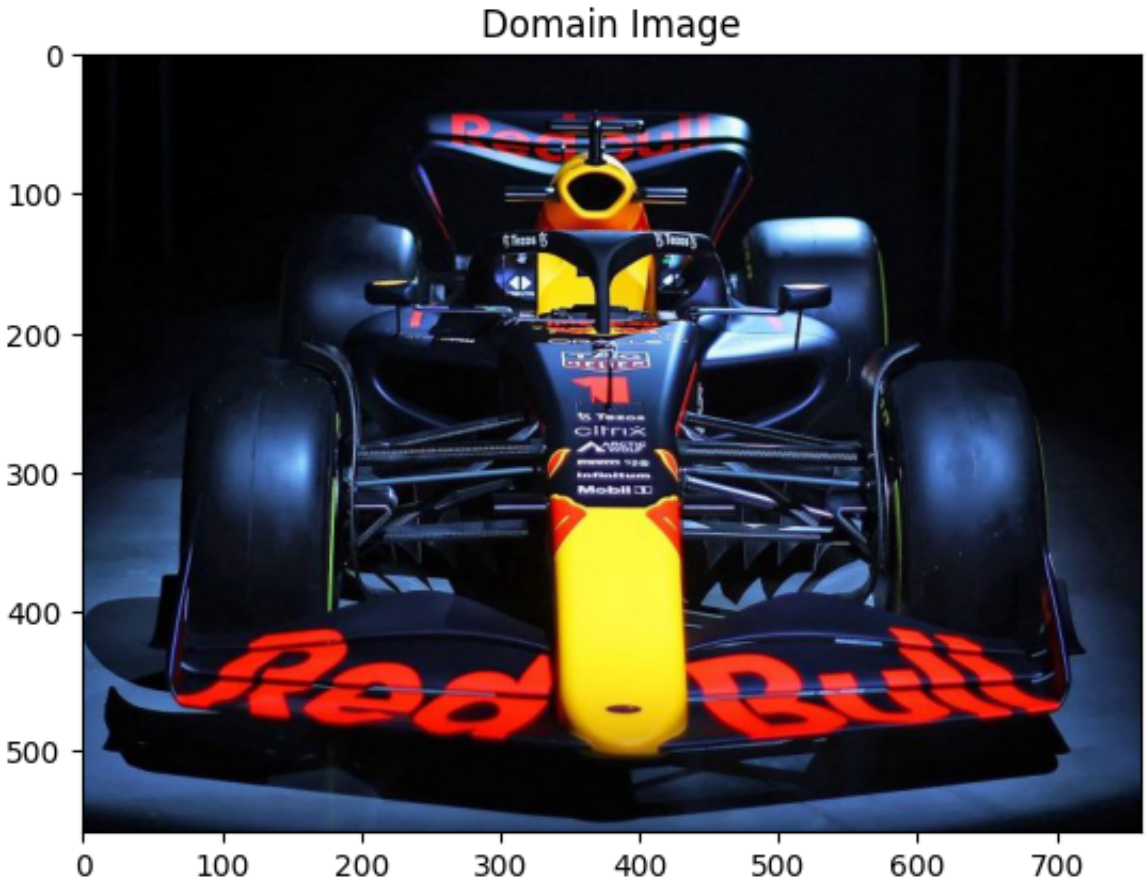

Domain Image
0
100
200
300
400
500
0
100
200
300
400
500
600
700

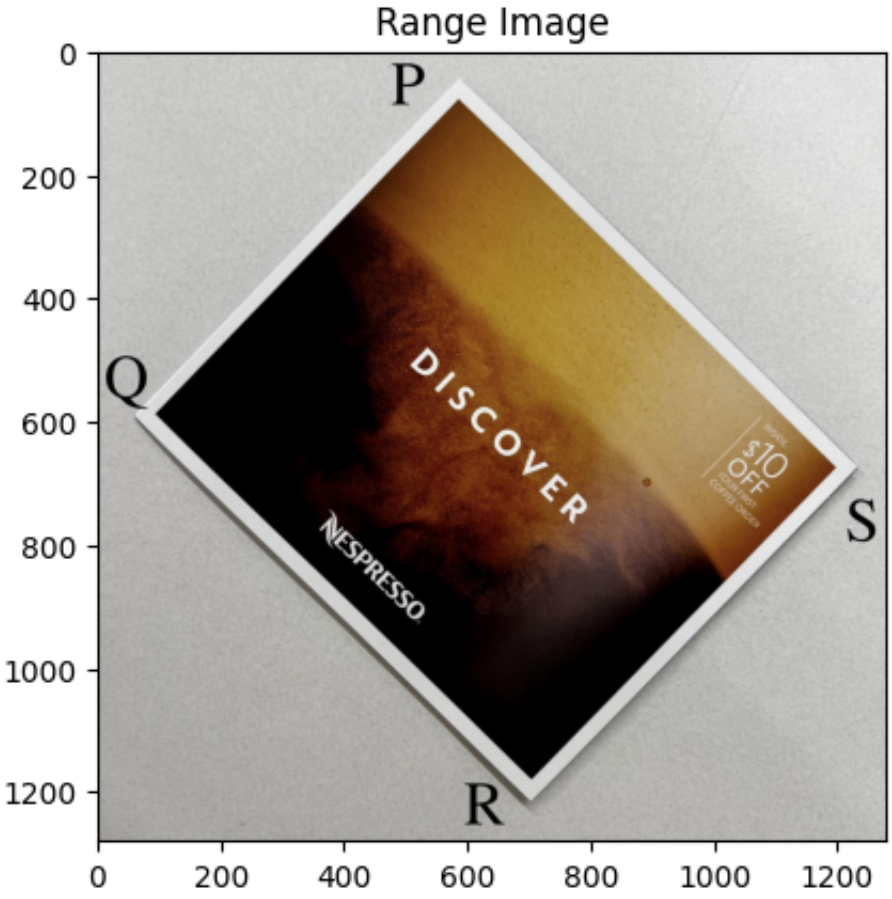

Range Image
P
S
Q
R
DISCOVER
NESPRESSO
$10
OFF
0
200
400
600
800
1000
1200

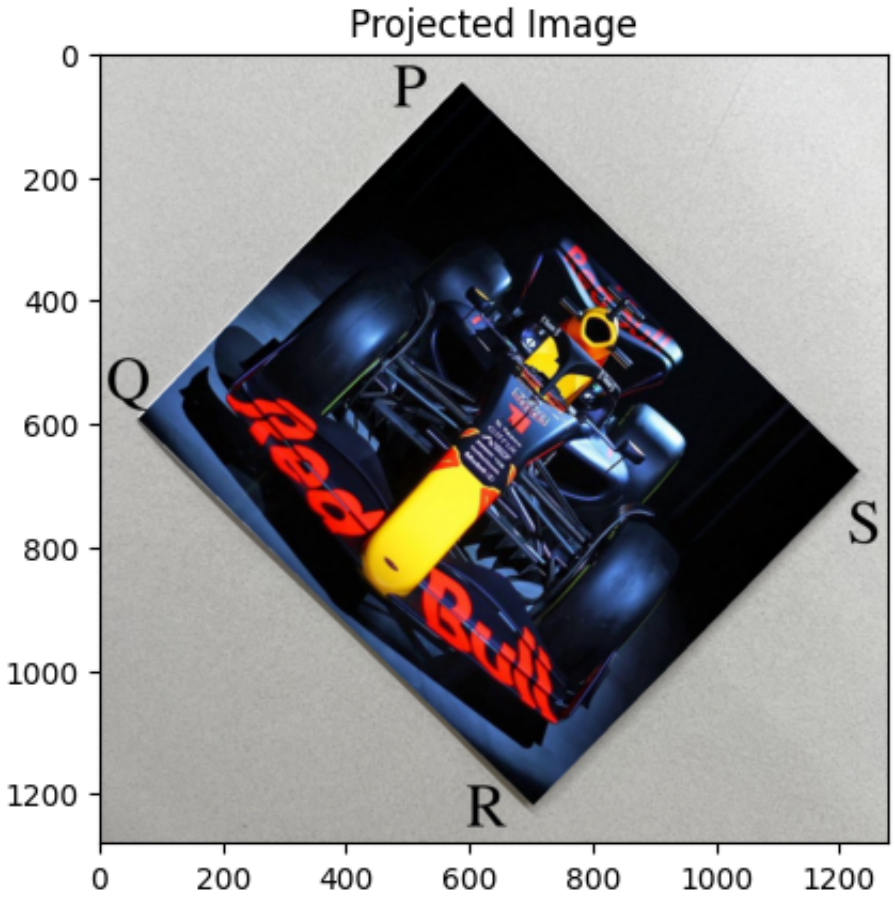


Card 3

In [71]:
```
a = np.array([[0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 1],
              [0, 558, 1, 0, 0, 0],
              [0, 0, 0, 0, 558, 1],
              [760, 0, 1, 0, 0, 0],
              [0, 0, 0, 760, 0, 1],
              [760, 558, 1, 0, 0, 0],
              [0, 0, 0, 760, 558, 1]])

b = np.array([587, 48, 62, 594, 1232, 678, 702, 1219])

H = findHomographyMatrixLeastSquares(a, b)

ROIImg = extractPQRS(card3.shape[0], card3.shape[1], [587, 48], [62, 594], [


newImg = inverseImgWarping(domainImg, np.copy(card3), ROIImg, H)


plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(card3)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task1.3_carToCard3.png')
```


Domain Image
0
100
200
300
400
500
0
100
200
300
400
500
600
700


Range Image
P
Q
S
R
DISCOVER
NESPRESSO
$10
OFF
0
200
400
600
800
1000
1200



The result we get from projecting the car image to card3 image using only affine transformations is clearly better than the other two. As we can see, lines PS and QR in Figs. 1a and 1b are not parallel, meaning that there are projective transformations involved. So using pure affine transformations couldn't get a good result. On the other hand, in Fig. 1c, lines PS and QR are parallel and so are PQ and SR, meaning that there are only affine transformations involved, thus, we can get a better result.

Task 2

Images used in task 2

In [3]:
```
pic1 = img.imread('./hw2images/pic1.png')
pic2 = img.imread('./hw2images/pic2.png')
pic3 = img.imread('./hw2images/pic3.png')
domainImg = img.imread('./hw2images/lily.png')

pic1 = cv2.cvtColor(pic1, cv2.COLOR_BGRA2BGR)
pic2 = cv2.cvtColor(pic2, cv2.COLOR_BGRA2BGR)
pic3 = cv2.cvtColor(pic3, cv2.COLOR_BGRA2BGR)

pic1 = cv2.resize(pic1, (848, 1088))
pic3 = cv2.resize(pic3, (848, 1088))
```

In [9]:
```
fig = plt.figure()
plt.imshow(pic1)
plt.axis('off')
plt.title('Fig. 2a')

fig = plt.figure()
plt.imshow(pic2)
plt.axis('off')
plt.title('Fig. 2b')


fig = plt.figure()
plt.imshow(pic3)
plt.axis('off')
plt.title('Fig. 2c')

lilyL = img.imread('./hw2images/lily_label.png')

fig = plt.figure()
plt.imshow(lilyL)
plt.axis('off')
plt.title('Fig. 2d')
```

Out[9]:
```
Text(0.5, 1.0, 'Fig. 2d')
```

Fig. 2a

Fig. 2b

Fig. 2c

Fig. 2d

Task 2.1

Step 1: Find coordinates of points P, Q, R and S in Figs 2a, 2b, 2c and A, B, C and D in Fig. 2d.

Step 2: Compute the homography matrix $\mathbf{H}$ as described at the beginning of this document.

Step 3: Extract the PQRS frame region in the range space.

Step 4: For every point (x', y') ($\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$ in $\mathbb{R}^3$, where $x' = \frac{x'_1}{x'_3}$ and $y' = \frac{x'_2}{x'_3}$) in the PQRS region, compute its correspondence point (x, y) ($\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ in $\mathbb{R}^3$, where $x = \frac{x_1}{x_3}$ and $y = \frac{x_2}{x_3}$) in the domain space using

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{H^{-1}} \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$$

and the pixel value in each color channel at (x', y') to the pixel value at (x, y) (Inverse Image Warping). If x or y is not a integer, using bilinear interpolation as described above to get an estimated pixel value at (x, y).

Projecting to picture 1

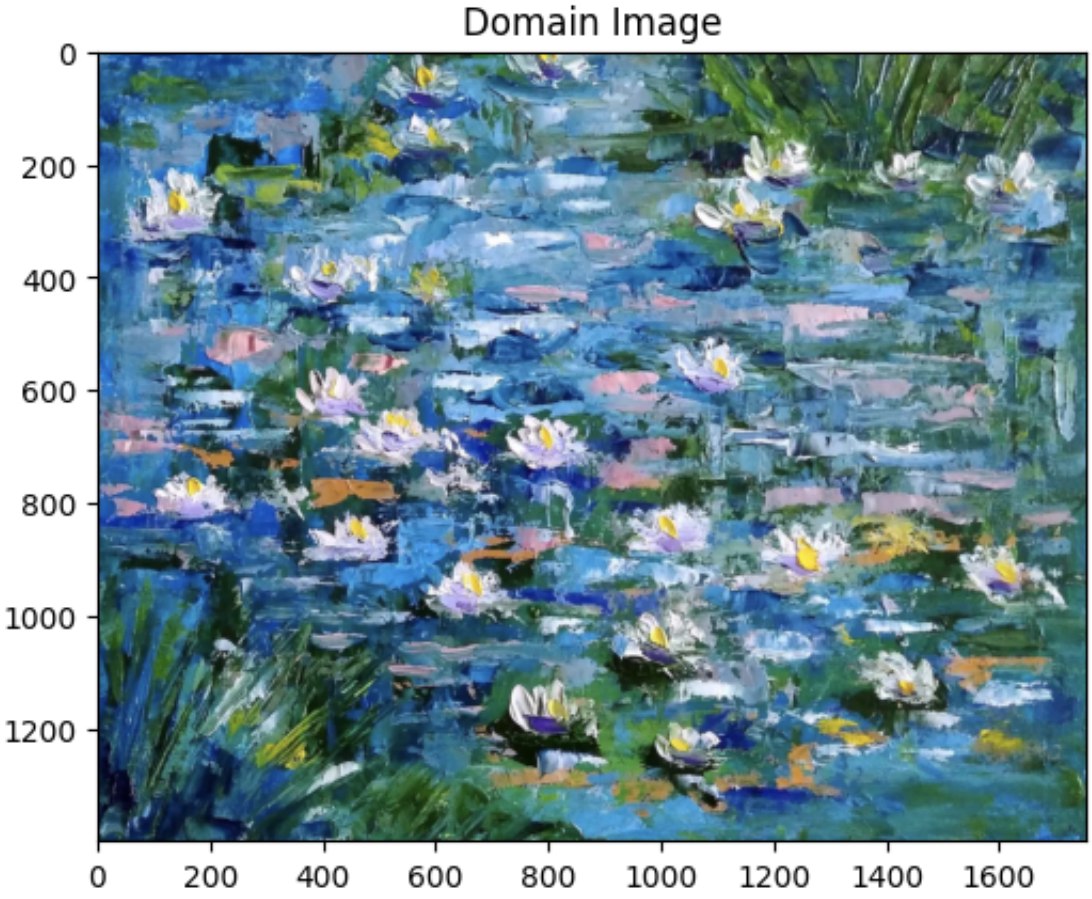
Correspondence points

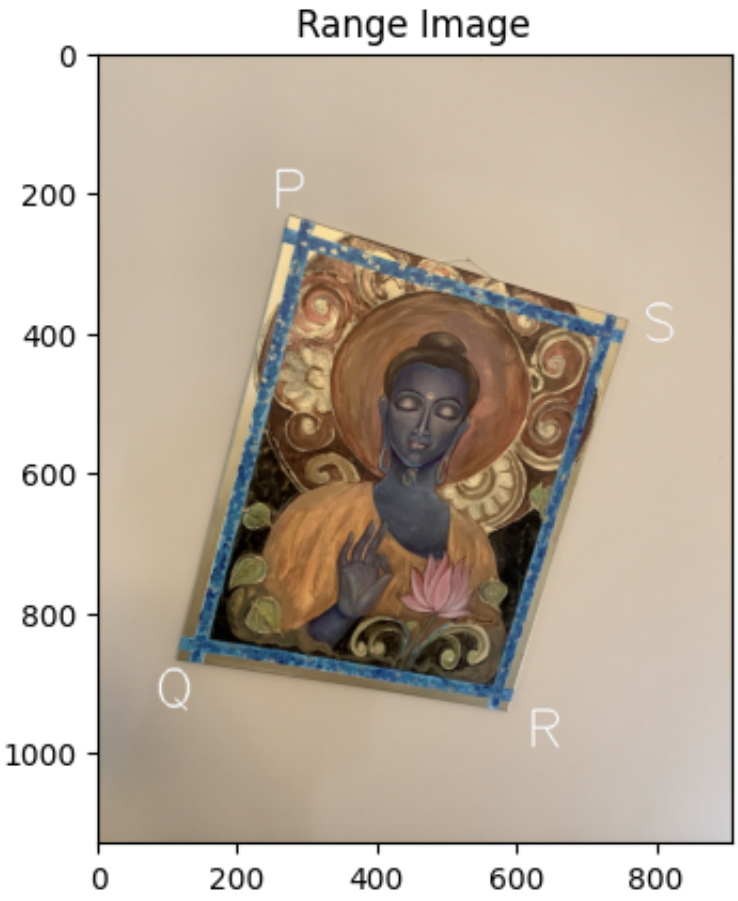
$$\begin{aligned}\text{Domain Space} &\rightarrow \text{Range Space}\\ A(417,191) &\rightarrow P(275,234)\\ B(417,1076) &\rightarrow Q(114,871)\\ C(1200,1076) &\rightarrow R(586,943)\\ D(1200,191) &\rightarrow S(758,386)\end{aligned}$$

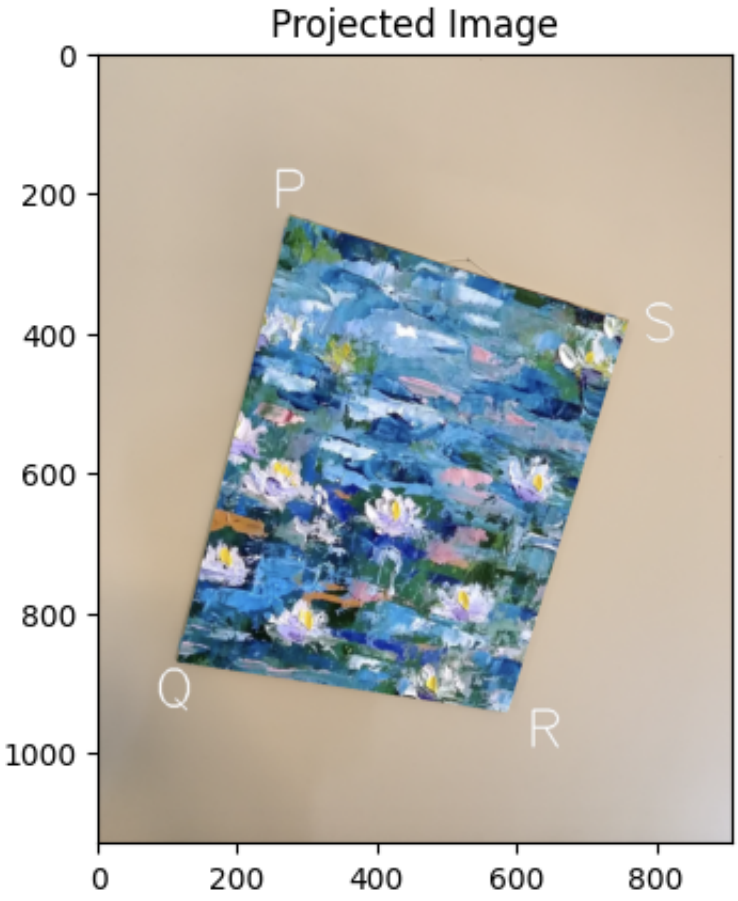
The 8 equations are:

$$\begin{aligned}417a_{11}+191a_{12}+t_x+0a_{21}+0a_{22}+0t_y-114675v_1-52525v_2&=275\\ 0a_{11}+0a_{12}+0t_x+417a_{21}+191a_{22}+t_y-97578v_1-44694v_2&=234\\ 417a_{11}+1076a_{12}+t_x+0a_{21}+0a_{22}+0t_y-47538v_1-122664v_2&=114\\ 0a_{11}+0a_{12}+0t_x+417a_{21}+1076a_{22}+t_y-362790v_1-937196v_2&=871\\ 1200a_{11}+1076a_{12}+t_x+0a_{21}+0a_{22}+0t_y-703200v_1-630536v_2&=586\\ 0a_{11}+0a_{12}+0t_x+1200a_{21}+1076a_{22}+t_y-1131600v_1-1014668v_2&=943\\ 1200a_{11}+191a_{12}+t_x+0a_{21}+0a_{22}+0t_y-909600v_1-144778v_2&=758\\ 0a_{11}+0a_{12}+0t_x+1200a_{21}+191a_{22}+t_y-463200v_1-73726v_2&=386\end{aligned}$$

In [73]:
```
a = np.array([[417, 191, 1, 0, 0, 0, -114675, -52525],
              [0, 0, 0, 417, 191, 1, -97578, -44694],
              [417, 1076, 1, 0, 0, 0, -47538, -122664],
              [0, 0, 0, 417, 1076, 1, -362790, -937196],
              [1200, 1076, 1, 0, 0, 0, -703200, -630536],
              [0, 0, 0, 1200, 1076, 1, -1131600, -1014668],
              [1200, 191, 1, 0, 0, 0, -909600, -144778],
              [0, 0, 0, 1200, 191, 1, -463200, -73726]])

b = np.array([275, 234, 114, 871, 586, 943, 758, 386])

H = findHomographyMatrix(a, b)


ROIImg = extractPQRS(pic1.shape[0], pic1.shape[1], [275, 234], [114, 871], [

newImg = inverseImgWarping(domainImg, np.copy(pic1), ROIImg, H)


plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(pic1)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task2.1_lilyToPic1.png')
```


Domain Image
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
1600


Range Image
0
200
400
600
800
1000
0
200
400
600
800
P
S
Q
R


Projected Image
P
S
Q
R
0
200
400
600
800
1000
0
200
400
600
800

Projecting to picture 2

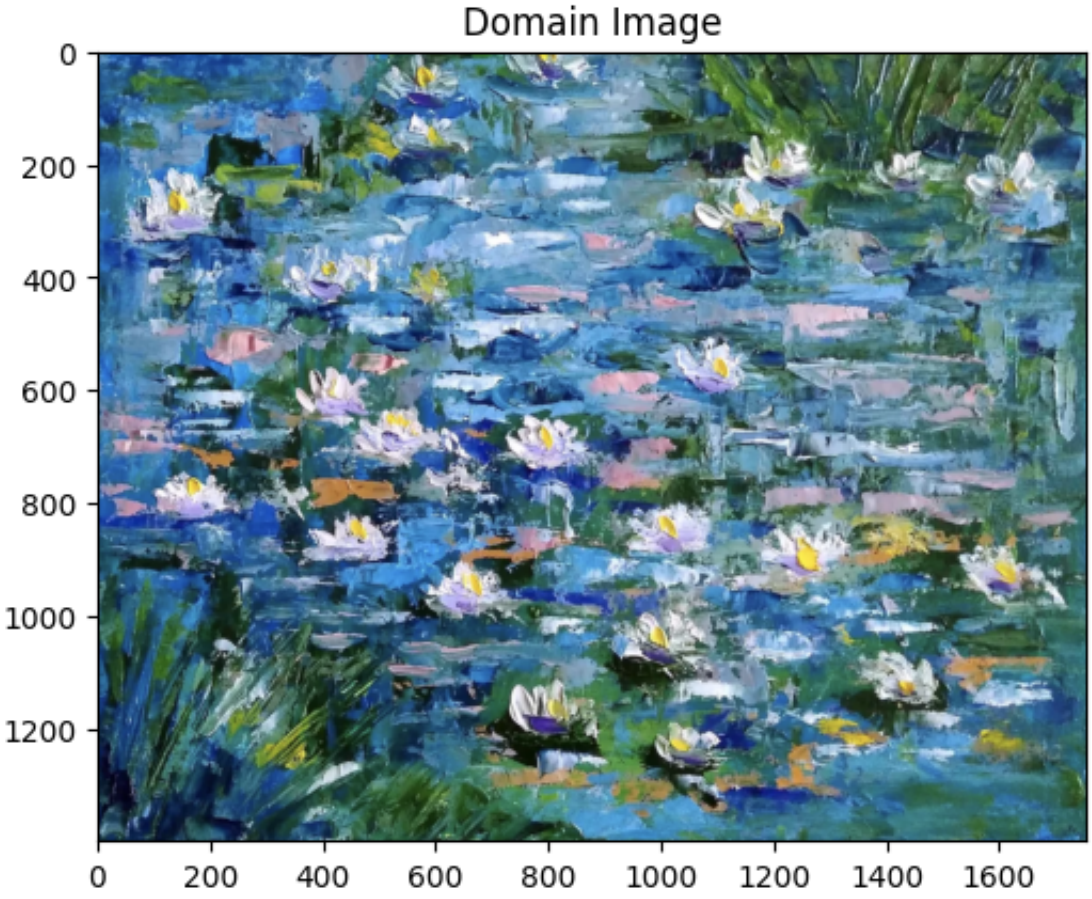
Correspondence points

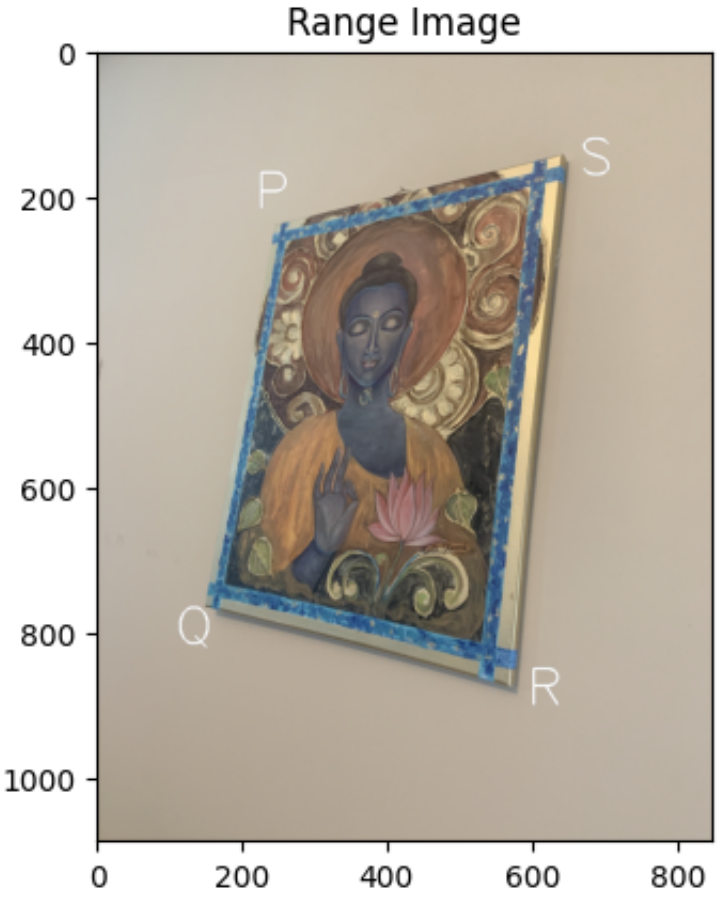
$$\begin{aligned}\text{Domain Space} &\rightarrow \text{Range Space}\\ A(417,191) &\rightarrow P(245,241)\\ B(417,1076) &\rightarrow Q(150,767)\\ C(1200,1076) &\rightarrow R(567,877)\\ D(1200,191) &\rightarrow S(639,147)\end{aligned}$$

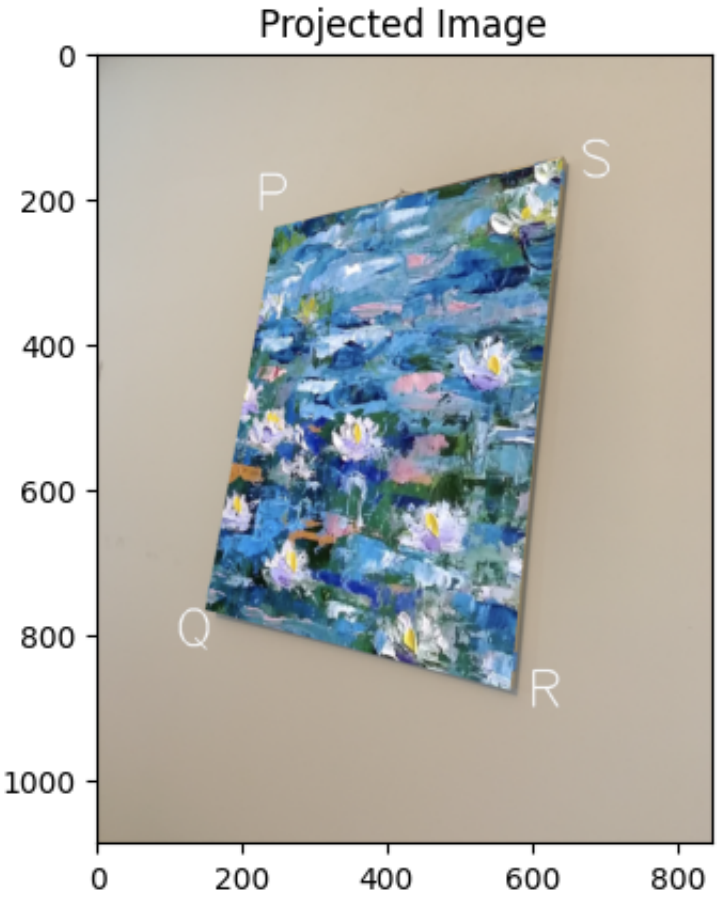
The 8 equations are:

$$\begin{aligned}417a_{11}+191a_{12}+t_x+0a_{21}+0a_{22}+0t_y-102165v_1-46795v_2&=245\\ 0a_{11}+0a_{12}+0t_x+417a_{21}+191a_{22}+t_y-100497v_1-46031v_2&=241\\ 417a_{11}+1076a_{12}+t_x+0a_{21}+0a_{22}+0t_y-62550v_1-161400v_2&=150\\ 0a_{11}+0a_{12}+0t_x+417a_{21}+1076a_{22}+t_y-319839v_1-825292v_2&=767\\ 1200a_{11}+1076a_{12}+t_x+0a_{21}+0a_{22}+0t_y-680400v_1-610092v_2&=567\\ 0a_{11}+0a_{12}+0t_x+1200a_{21}+1076a_{22}+t_y-1052400v_1-943652v_2&=877\\ 1200a_{11}+191a_{12}+t_x+0a_{21}+0a_{22}+0t_y-766800v_1-122049v_2&=639\\ 0a_{11}+0a_{12}+0t_x+1200a_{21}+191a_{22}+t_y-176400v_1-28077v_2&=147\end{aligned}$$

In [74]:
```
a = np.array([[417, 191, 1, 0, 0, 0, -102165, -46795],
              [0, 0, 0, 417, 191, 1, -100497, -46031],
              [417, 1076, 1, 0, 0, 0, -62550, -161400],
              [0, 0, 0, 417, 1076, 1, -319839, -825292],
              [1200, 1076, 1, 0, 0, 0, -680400, -610092],
              [0, 0, 0, 1200, 1076, 1, -1052400, -943652],
              [1200, 191, 1, 0, 0, 0, -766800, -122049],
              [0, 0, 0, 1200, 191, 1, -176400, -28077]])

b = np.array([245, 241, 150, 767, 567, 877, 639, 147])

H = findHomographyMatrix(a, b)

ROIImg = extractPQRS(pic2.shape[0], pic2.shape[1], [245, 241], [150, 767], [

newImg = inverseImgWarping(domainImg, np.copy(pic2), ROIImg, H)



plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(pic2)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task2.1_lilyToPic2.png')
```


Domain Image
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
1600


Range Image
0
200
400
600
800
1000
0
200
400
600
800
P
S
Q
R


Projected Image
P
S
Q
R
0
200
400
600
800
1000
0
200
400
600
800

Projecting to picture 3

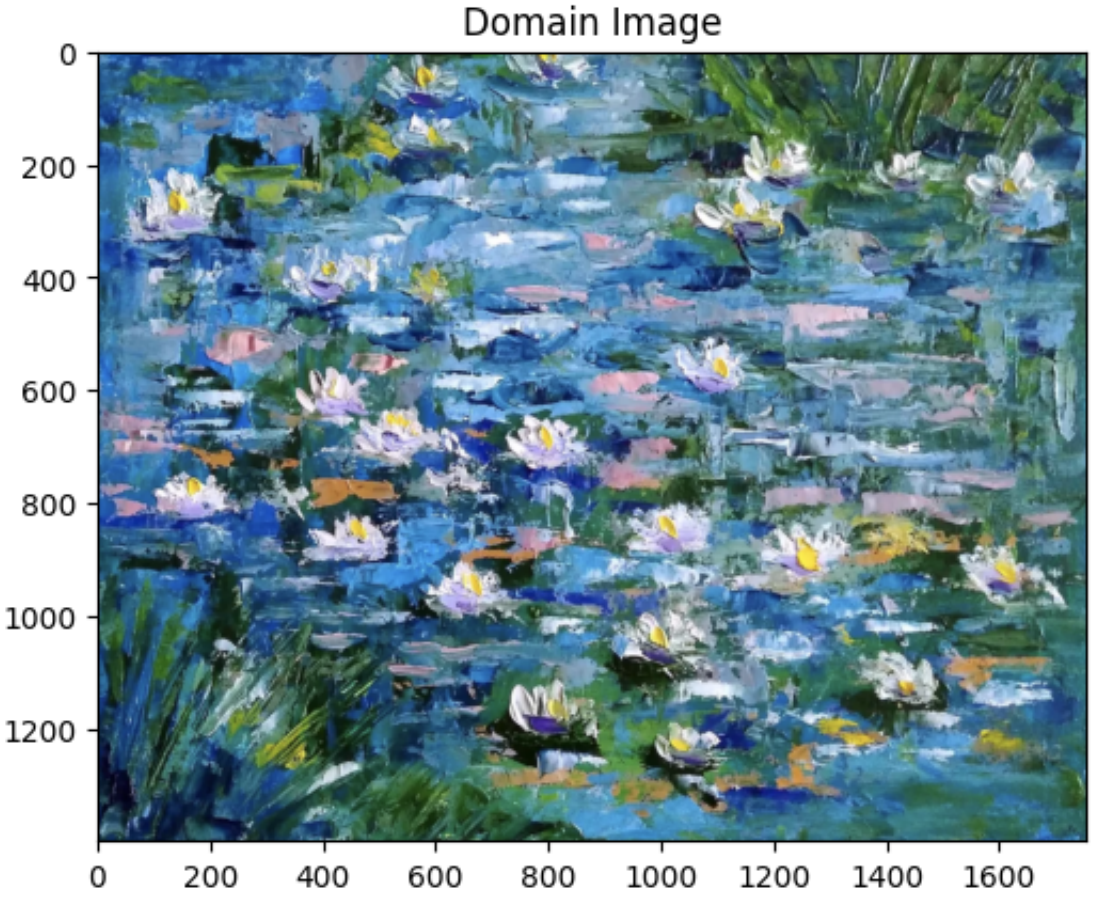
Correspondence points

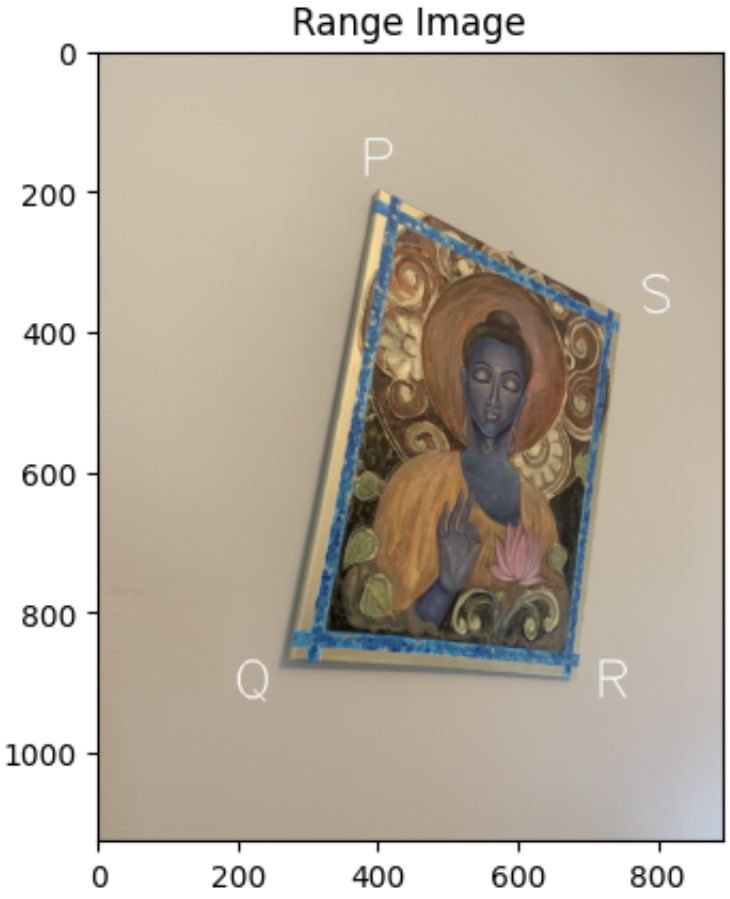
$$\begin{aligned}\text{Domain space} &\rightarrow \text{Range space}\\ A(417,191) &\rightarrow P(405,201)\\ B(417,1076) &\rightarrow Q(278,870)\\ C(1200,1076) &\rightarrow R(683,896)\\ D(1200,191) &\rightarrow S(746,377)\end{aligned}$$

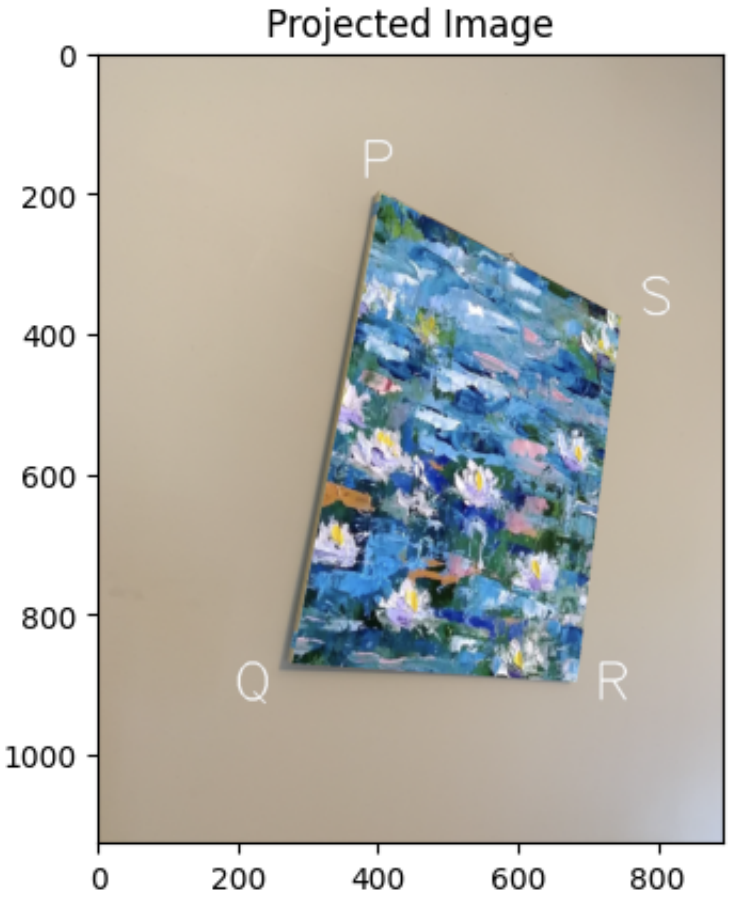
The 8 equations are:

$$\begin{aligned}417a_{11}+191a_{12}+t_x+0a_{21}+0a_{22}+0t_y-168885v_1-77355v_2&=405\\ 0a_{11}+0a_{12}+0t_x+417a_{21}+191a_{22}+t_y-83817v_1-38391v_2&=201\\ 417a_{11}+1076a_{12}+t_x+0a_{21}+0a_{22}+0t_y-115926v_1-299128v_2&=278\\ 0a_{11}+0a_{12}+0t_x+417a_{21}+1076a_{22}+t_y-362790v_1-936120v_2&=870\\ 1200a_{11}+1076a_{12}+t_x+0a_{21}+0a_{22}+0t_y-819600v_1-734908v_2&=683\\ 0a_{11}+0a_{12}+0t_x+1200a_{21}+1076a_{22}+t_y-1075200v_1-964096v_2&=896\\ 1200a_{11}+191a_{12}+t_x+0a_{21}+0a_{22}+0t_y-895200v_1-142486v_2&=746\\ 0a_{11}+0a_{12}+0t_x+1200a_{21}+191a_{22}+t_y-452400v_1-72007v_2&=377\end{aligned}$$

In [75]:
```
a = np.array([[417, 191, 1, 0, 0, 0, -168885, -77355],
              [0, 0, 0, 417, 191, 1, -83817, -38391],
              [417, 1076, 1, 0, 0, 0, -115926, -299128],
              [0, 0, 0, 417, 1076, 1, -362790, -936120],
              [1200, 1076, 1, 0, 0, 0, -819600, -734908],
              [0, 0, 0, 1200, 1076, 1, -1075200, -964096],
              [1200, 191, 1, 0, 0, 0, -895200, -142486],
              [0, 0, 0, 1200, 191, 1, -452400, -72007]])
b = np.array([405, 201, 278, 870, 683, 896, 746, 377])

H = findHomographyMatrix(a, b)


ROIImg = extractPQRS(pic3.shape[0], pic3.shape[1], [405, 201], [278, 870], [


newImg = inverseImgWarping(domainImg, np.copy(pic3), ROIImg, H)



plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(pic3)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task2.1_lilyToPic3.png')
```


Domain Image
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
1600


Range Image
P
S
Q
R
0
200
400
600
800
1000
0
200
400
600
800


Projected Image
P
S
Q
R
0
200
400
600
800
1000
0
200
400
600
800

Task 2.2

Step 1: Find the homography matrix $\mathbf{H_{ab}}$ using the logic described above and the correspondence points in this case are corners (P, Q, R and S) of Figs. 2a and 2b. Similarly, we can find the homography matrix $\mathbf{H_{bc}}$ between Figs. 2b and 2c.

Step 2: Compute the product of $\mathbf{H_{bc}}$ and $\mathbf{H_{ab}}$.

Step 3: Extract ROI in Fig. , which is the entire image.

Step 4: For each point in ROI (Fig. 2c), use inverse image wraping to find its correspondence point in Fig. 2a. If coordinates of the calculated point are not integers, using bilinear interpolation as described above to get an estimated pixel value.

Correspondence points in Figs. 2a and 2b

$$\begin{gathered}\text{Fig. 2a} \rightarrow \text{Fig. 2b} \\ (275, 234) \rightarrow (245, 241) \\ (114, 871) \rightarrow (150, 767) \\ (586, 943) \rightarrow (567, 877) \\ (758, 386) \rightarrow (639, 147)\end{gathered}$$

The 8 equations are:

$$\begin{gathered}275a_{11} + 234a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 67375v_1 - 57330v_2 = 245 \\ 0a_{11} + 0a_{12} + 0t_x + 275a_{21} + 234a_{22} + t_y - 66275v_1 - 56394v_2 = 241 \\ 114a_{11} + 871a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 17100v_1 - 130650v_2 = 150 \\ 0a_{11} + 0a_{12} + 0t_x + 114a_{21} + 871a_{22} + t_y - 87438v_1 - 668057v_2 = 767 \\ 586a_{11} + 943a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 332262v_1 - 534681v_2 = 567 \\ 0a_{11} + 0a_{12} + 0t_x + 586a_{21} + 943a_{22} + t_y - 513922v_1 - 827011v_2 = 877 \\ 758a_{11} + 386a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 484362v_1 - 246654v_2 = 639 \\ 0a_{11} + 0a_{12} + 0t_x + 758a_{21} + 386a_{22} + t_y - 111426v_1 - 56742v_2 = 147\end{gathered}$$

In [76]:
```
a = np.array([[275, 234, 1, 0, 0, 0, -67375, -57330],
              [0, 0, 0, 275, 234, 1, -66275, -56394],
              [114, 871, 1, 0, 0, 0, -17100, -130650],
              [0, 0, 0, 114, 871, 1, -87438, -668057],
              [586, 943, 1, 0, 0, 0, -332262, -534681],
              [0, 0, 0, 586, 943, 1, -513922, -827011],
              [758, 386, 1, 0, 0, 0, -484362, -246654],
              [0, 0, 0, 758, 386, 1, -111426, -56742]])

b = np.array([245, 241, 150, 767, 567, 877, 639, 147])

Hab = findHomographyMatrix(a, b)
```

Correspondence points in Figs. 2b and 2c

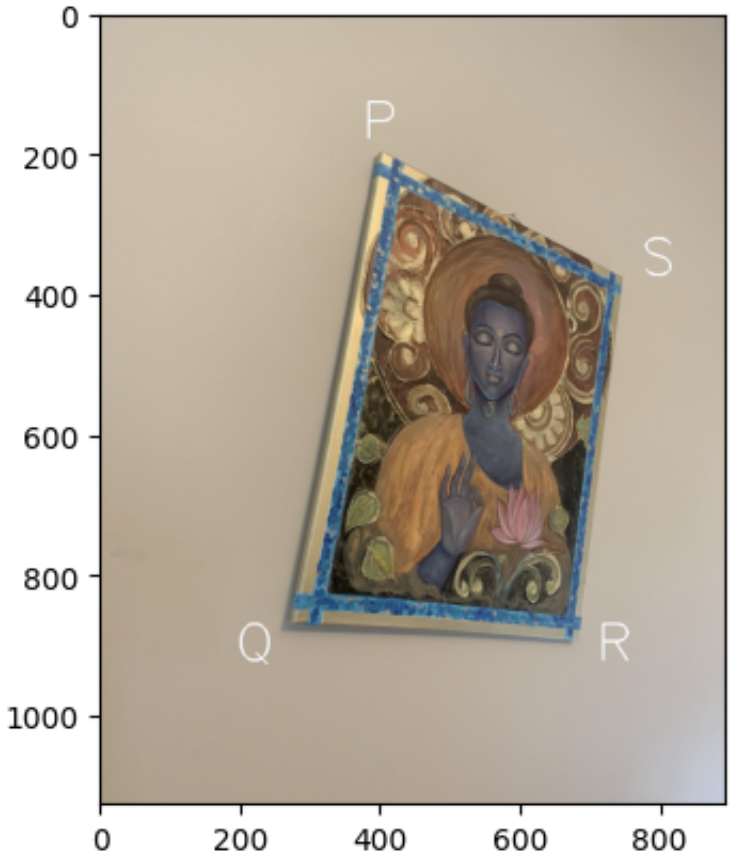
$$\begin{aligned}\text{Fig. 2b} &\rightarrow \text{Fig. 2c}\\ (245, 241) &\rightarrow (405, 201)\\ (150, 767) &\rightarrow (278, 870)\\ (567, 877) &\rightarrow (683, 896)\\ (639, 147) &\rightarrow (746, 377)\end{aligned}$$

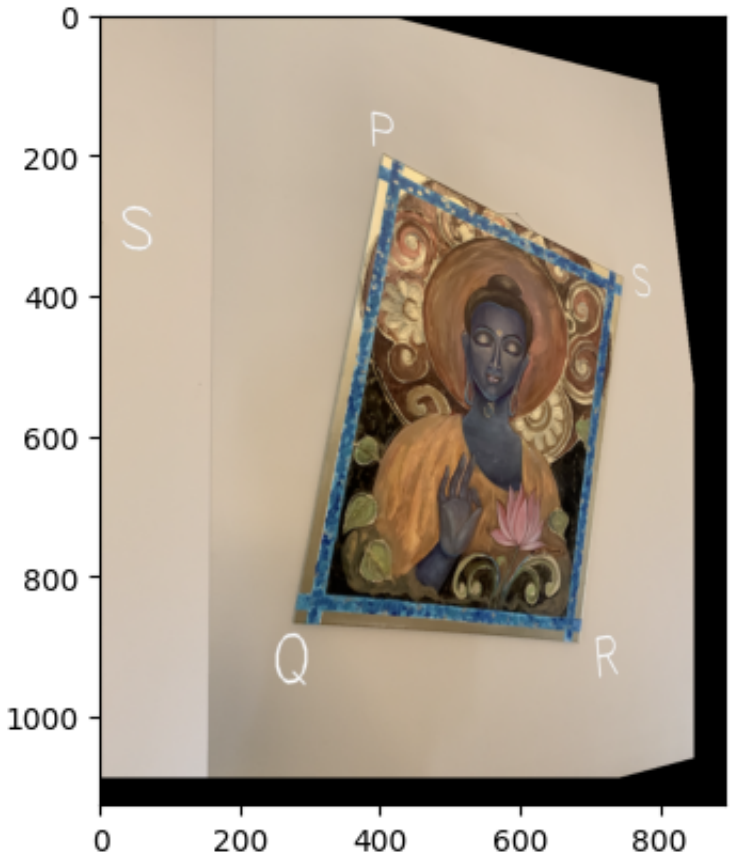
The 8 equations are:

$$\begin{aligned}245a_{11} + 241a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 99225v_1 - 97605v_2 &= 405\\ 0a_{11} + 0a_{12} + 0t_x + 245a_{21} + 241a_{22} + t_y - 49245v_1 - 48441v_2 &= 201\\ 150a_{11} + 767a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 41700v_1 - 213226v_2 &= 278\\ 0a_{11} + 0a_{12} + 0t_x + 150a_{21} + 767a_{22} + t_y - 130500v_1 - 667290v_2 &= 870\\ 567a_{11} + 877a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 387261v_1 - 598991v_2 &= 683\\ 0a_{11} + 0a_{12} + 0t_x + 567a_{21} + 877a_{22} + t_y - 508032v_1 - 785792v_2 &= 896\\ 639a_{11} + 147a_{12} + t_x + 0a_{21} + 0a_{22} + 0t_y - 476694v_1 - 109662v_2 &= 746\\ 0a_{11} + 0a_{12} + 0t_x + 639a_{21} + 147a_{22} + t_y - 240903v_1 - 55419v_2 &= 377\end{aligned}$$

In [77]:
```
a = np.array([[245, 241, 1, 0, 0, 0, -99225, -97605],
              [0, 0, 0, 245, 241, 1, -49245, -48441],
              [150, 767, 1, 0, 0, 0, -41700, -213226],
              [0, 0, 0, 150, 767, 1, -130500, -667290],
              [567, 877, 1, 0, 0, 0, -387261, -598991],
              [0, 0, 0, 567, 877, 1, -508032, -785792],
              [639, 147, 1, 0, 0, 0, -476694, -109662],
              [0, 0, 0, 639, 147, 1, -240903, -55419]])
b = np.array([405, 201, 278, 870, 683, 896, 746, 377])

Hbc = findHomographyMatrix(a, b)
```

In [83]:
```
H = np.matmul(Hbc, Hab)


ROIImg = extractPQRS(pic3.shape[0], pic3.shape[1], [0, 0], [0, 1087], [847,

ans = np.zeros((pic3.shape[0], pic3.shape[1], 3), dtype='float')

newImg = inverseImgWarping(pic1, ans, ROIImg, H)

fig = plt.figure()
plt.imshow(pic3)

fig = plt.figure()
plt.imshow(newImg)
plt.savefig('./hw2images/task2.2_pic1ToPic3.png')
```





Task 2.3

Steps

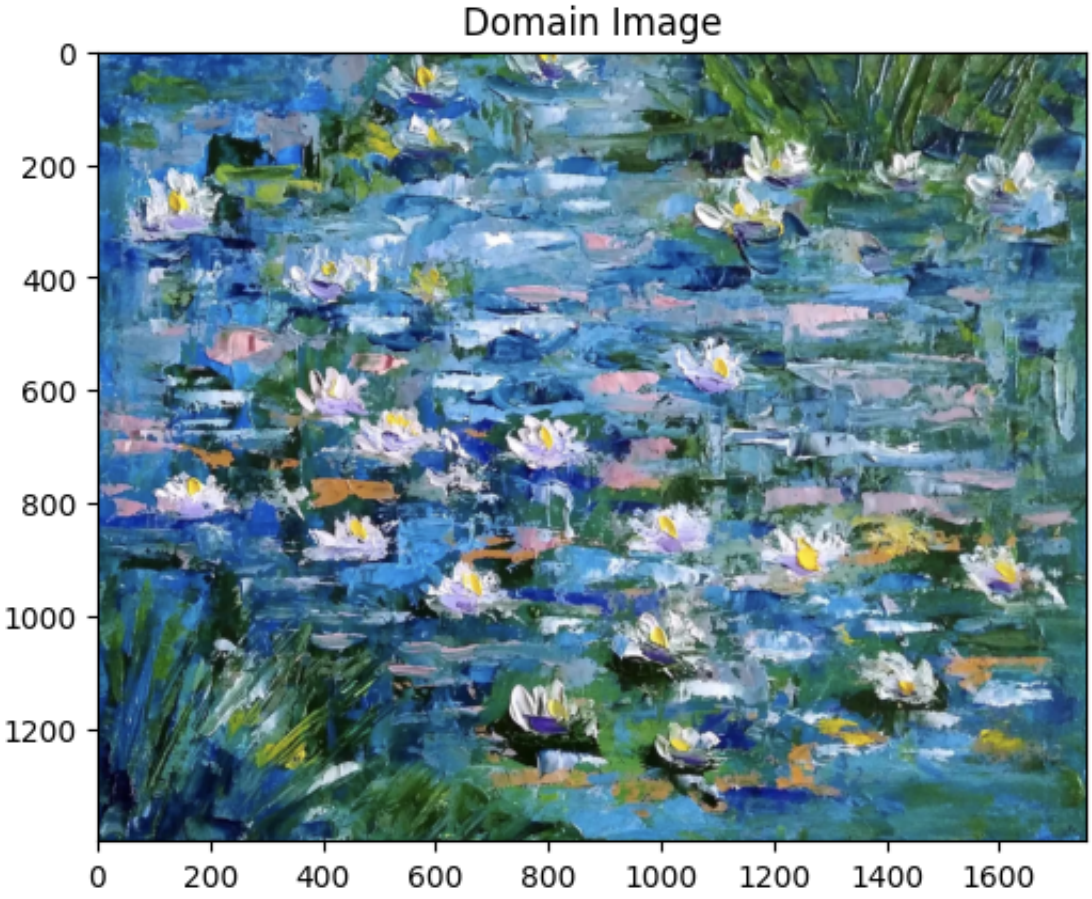
Step 1: Find correspondence points and compute the homography matrix H using the logic described at the beginning of the document. However, since we now have an overdetermined system (8 linear equations and 6 unknowns), we need to use least square method to solve the overdetermined system.

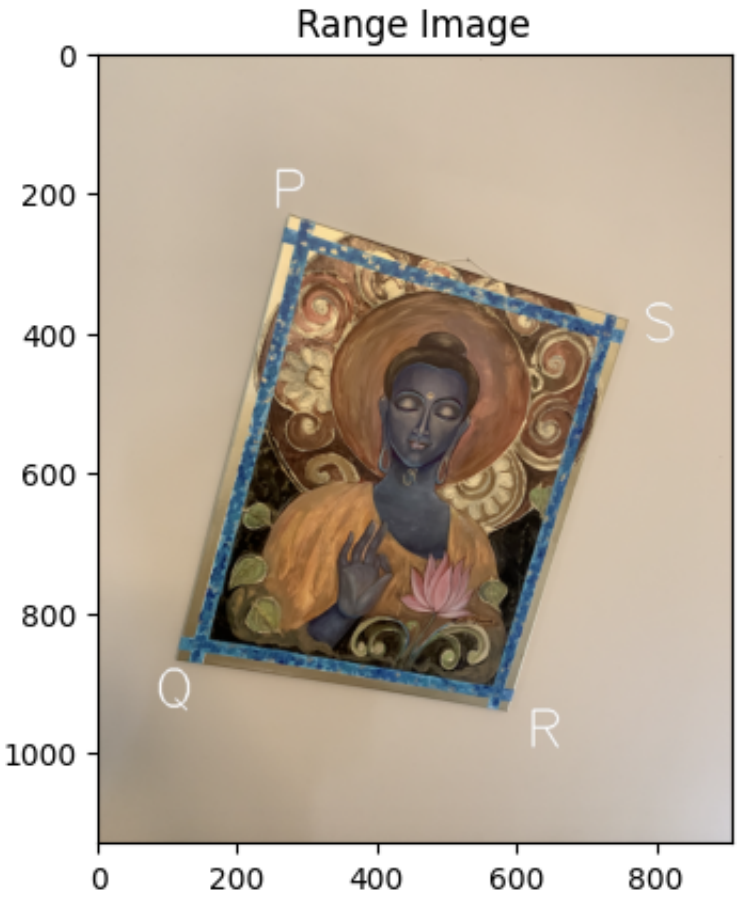
Step 2: Extract the PQRS frame region in the range space.

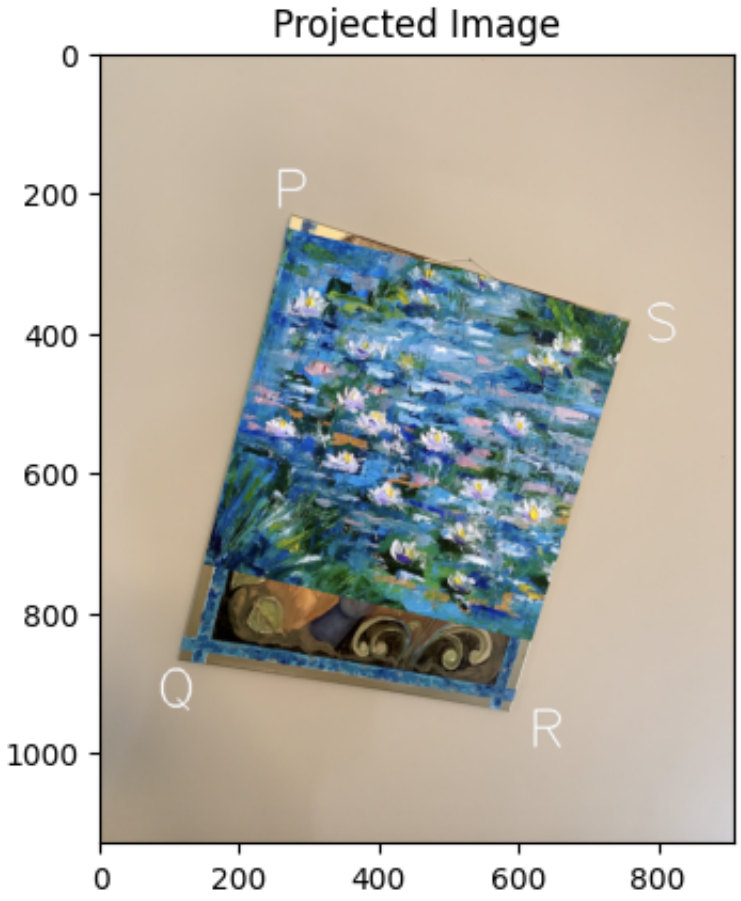
Step 3: Perform inverse image warping to project the entire lily image to the PQRS frame region. Use bilinear interpolation to get an estimated pixel value if the caculated coordinates are not integers.

Picture 1

In [84]:
```
a = np.array([[0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 1],
              [0, 1753, 1, 0, 0, 0],
              [0, 0, 0, 0, 1753, 1],
              [1399, 1753, 1, 0, 0, 0],
              [0, 0, 0, 1399, 1753, 1],
              [1399, 0, 1, 0, 0, 0],
              [0, 0, 0, 1399, 0, 1]])

b = np.array([275, 234, 114, 871, 586, 943, 758, 386])

H = findHomographyMatrixLeastSquares(a, b)


ROIImg = extractPQRS(pic1.shape[0], pic1.shape[1], [275, 234], [114, 871], [


newImg = inverseImgWarping(domainImg, np.copy(pic1), ROIImg, H)



plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(pic1)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task2.3_lilyToPic1.png')
```

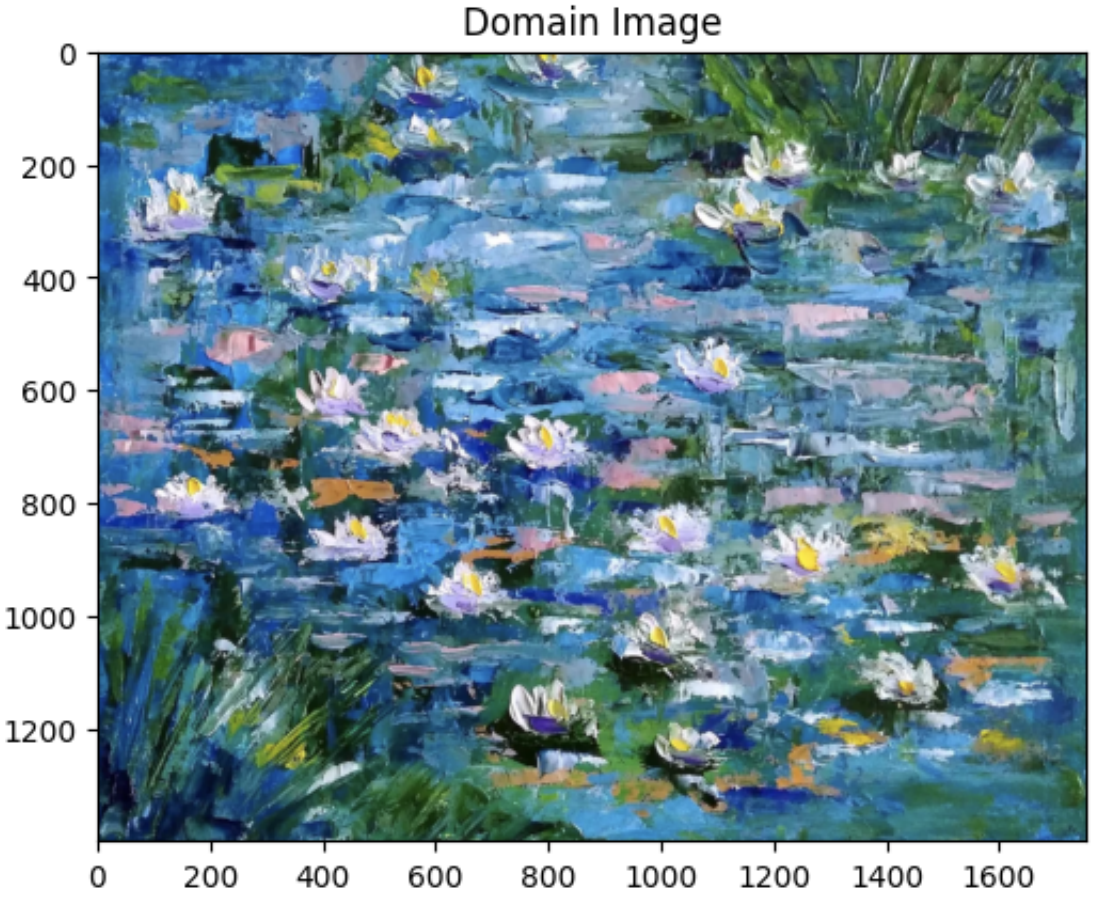

Domain Image
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
1600

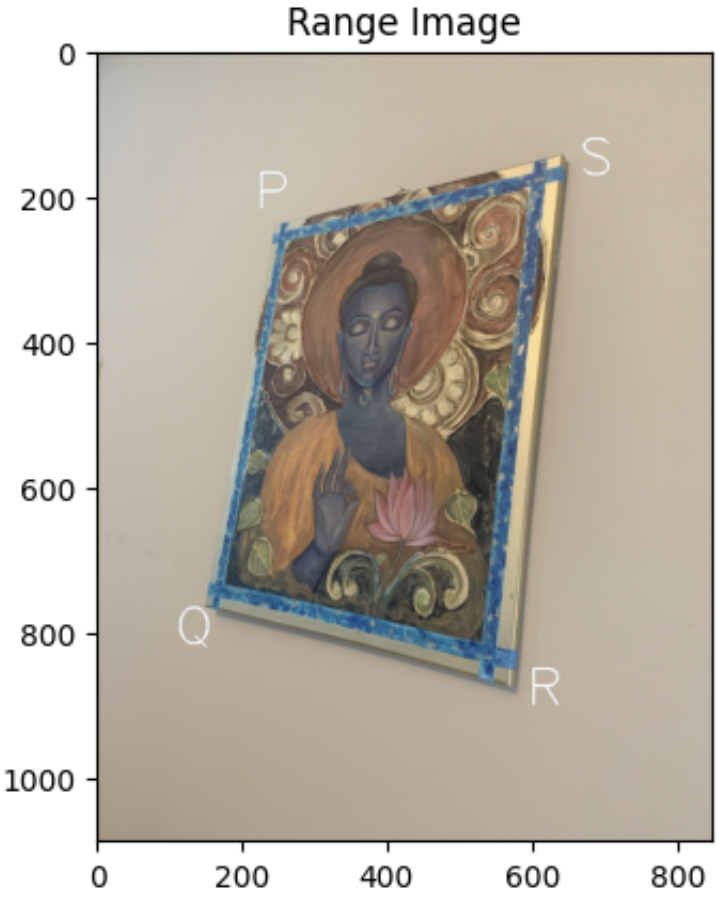

Range Image
P
S
Q
R
0
200
400
600
800
1000
0
200
400
600
800

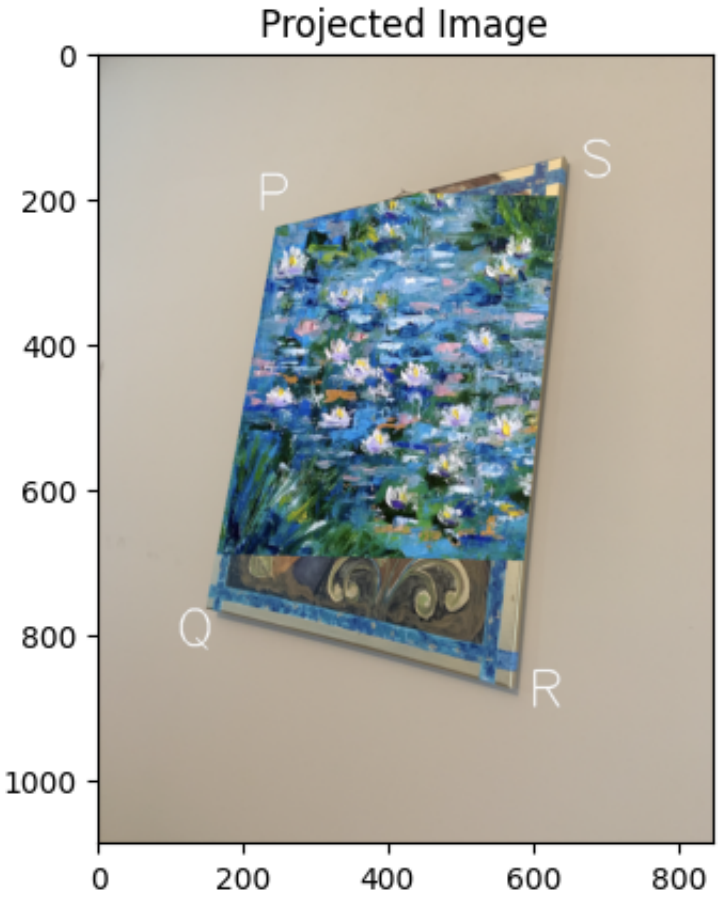


Picture 2

In [85]:
```
a = np.array([[0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 1],
              [0, 1753, 1, 0, 0, 0],
              [0, 0, 0, 0, 1753, 1],
              [1399, 1753, 1, 0, 0, 0],
              [0, 0, 0, 1399, 1753, 1],
              [1399, 0, 1, 0, 0, 0],
              [0, 0, 0, 1399, 0, 1]
             ])

b = np.array([245, 241, 150, 767, 567, 877, 639, 147])


H = findHomographyMatrixLeastSquares(a, b)


ROIImg = extractPQRS(pic2.shape[0], pic2.shape[1], [245, 241], [150, 767], [


newImg = inverseImgWarping(domainImg, np.copy(pic2), ROIImg, H)


plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(pic2)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task2.3_lilyToPic2.png')
```

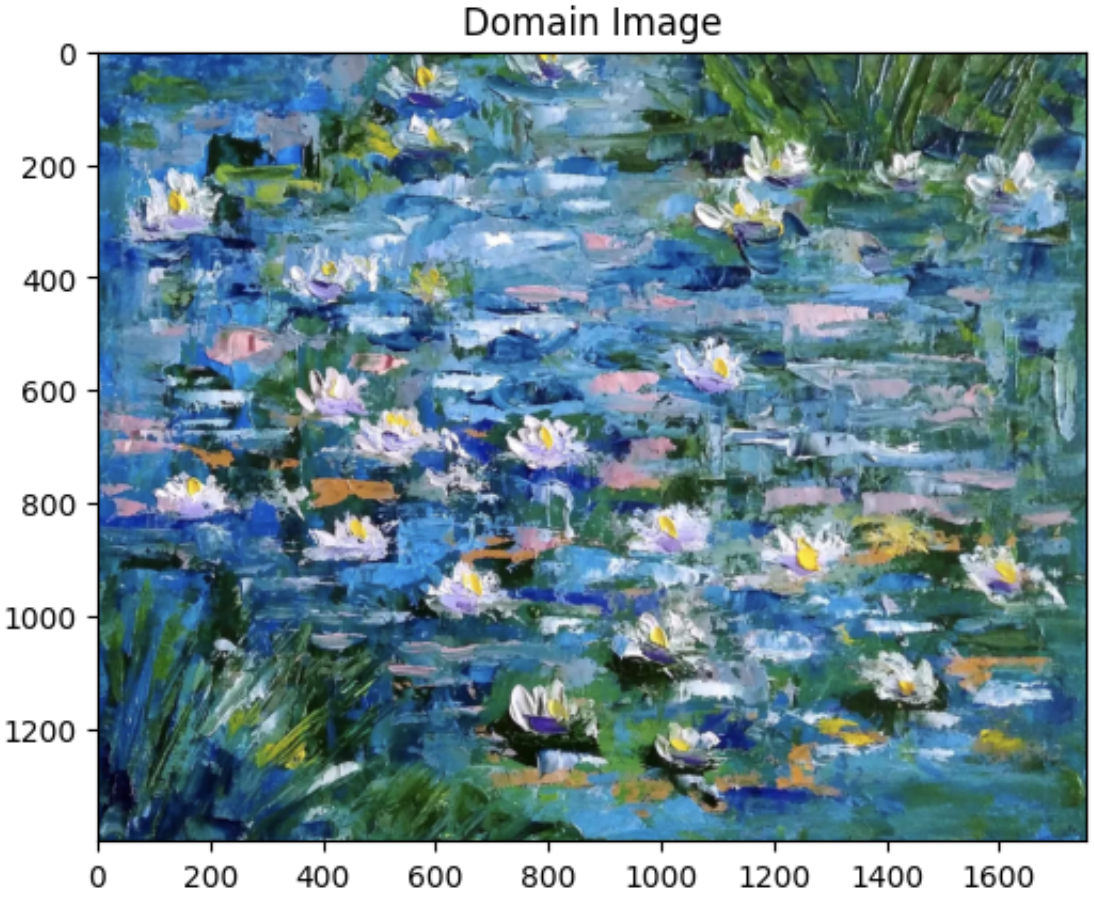

Domain Image
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
1600

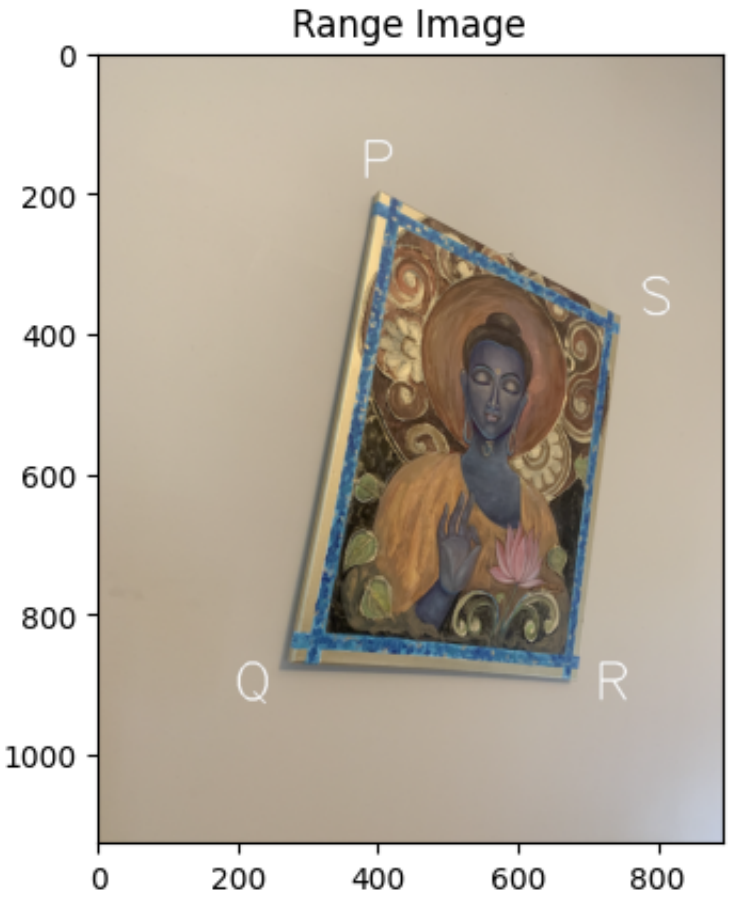

Range Image
0
200
400
600
800
1000
0
200
400
600
800
P
S
Q
R

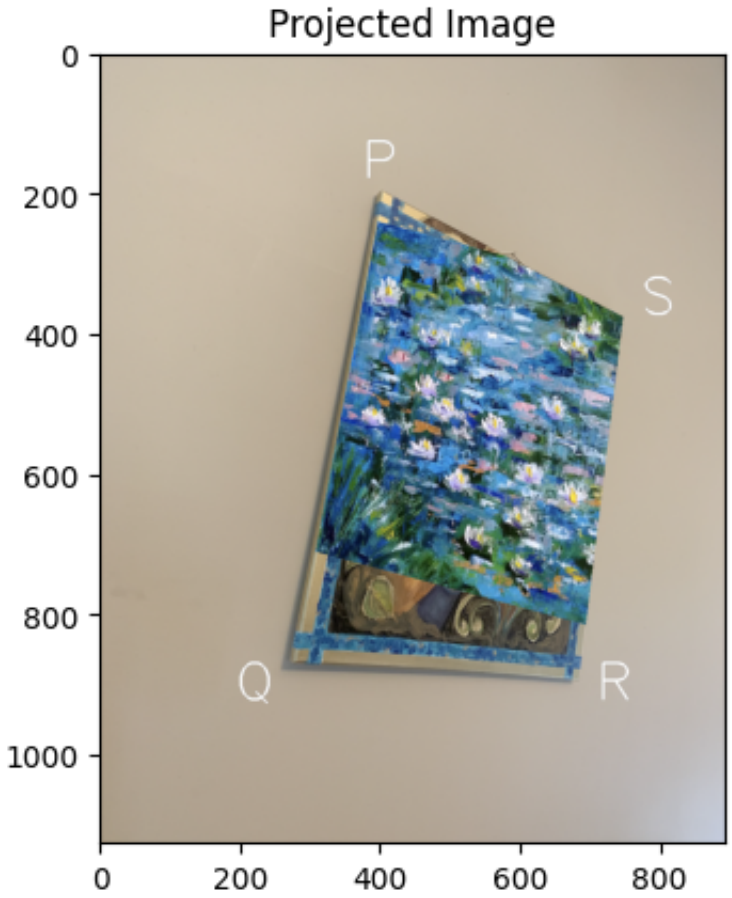


Picture 3

In [86]:
```
a = np.array([[0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 1],
              [0, 1753, 1, 0, 0, 0],
              [0, 0, 0, 0, 1753, 1],
              [1399, 1753, 1, 0, 0, 0],
              [0, 0, 0, 1399, 1753, 1],
              [1399, 0, 1, 0, 0, 0],
              [0, 0, 0, 1399, 0, 1]
             ])
b = np.array([405, 201, 278, 870, 683, 896, 746, 377])

H = findHomographyMatrixLeastSquares(a, b)

ROIImg = extractPQRS(pic3.shape[0], pic3.shape[1], [405, 201], [278, 870], [


newImg = inverseImgWarping(domainImg, np.copy(pic3), ROIImg, H)



plt.imshow(domainImg)
plt.title('Domain Image')

fig = plt.figure()
plt.imshow(pic3)
plt.title('Range Image')

fig = plt.figure()
plt.imshow(newImg)
plt.title('Projected Image')
plt.savefig('./hw2images/task2.3_lilyToPic3.png')
```


Domain Image
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400
1600


Range Image
P
S
Q
R
0
200
400
600
800
1000
0
200
400
600
800



The 3 results we get are all bad. As we can see from Figs. 2a, 2b and 2c, lines PS and QR are not parallel, meaning that there are projective transformations involved. Thus, using pure affine transformations couldn't produce a good result.

References

[1] Ron, A., 2010. [online] Pages.cs.wisc.edu. Available at: https://pages.cs.wisc.edu/~amos/412/lecture-notes/lecture17.pdf [Accessed 7 September 2022].

[2] En.wikipedia.org. 2022. Conic section - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Conic_section [Accessed 31 August 2022].