ECE 661 Homework 2

Suyash Ail
sail@purdue.edu

September 11, 2020

The task of this homework is to find homography between two images using 4 point correspondence. The homography matrix is then used to project one image onto the other.

1 Logic

Let $\vec{x}$ be the homogeneous representation of a point in the domain. Let $\vec{x}'$ be the homogeneous representation of the point in the projective space (range). The relationship between the two points is represented by a non-singular 3x3 matrix H which is called the homography matrix.

$$\vec{x}' = \mathrm{H}\vec{x} \tag{1}$$

H is a linear mapping from one homogeneous coordinate to another.

$$\mathrm{H} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$$

The matrix H is

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

Since the Homography matrix is homogeneous, the information is in the ratios, hence h_{33}=1.

Let us assume a point in the domain $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ and a corresponding point in the range $\vec{x}' = \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix}$.

Hence

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

$$x_1' = h_{11}x_1 + h_{12}x_2 + h_{13}x_3$$
$$x_2' = h_{21}x_1 + h_{22}x_2 + h_{23}x_3$$
$$x_3' = h_{31}x_1 + h_{32}x_2 + x_3$$

Since the information is in the ratios, we get the x and y coordinates of the transformed points as

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x_1 + h_{12}x_2 + h_{13}x_3}{h_{31}x_1 + h_{32}x_2 + x_3}$$

$$y' = \frac{x'_2}{x'_3} = \frac{h_{21}x_1 + h_{22}x_2 + h_{23}x_3}{h_{31}x_1 + h_{32}x_2 + x_3}$$

Dividing the right hand side by x_3 gives

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

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

$$\therefore h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' - x' = 0 \tag{2}$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' + h_{32}yy' + y' = 0 \tag{3}$$

Here we have two equations with 8 unknowns. We see that one pair of corresponding points gives two equations. Hence we need 4 such points.
Rewriting the equation in matrix form we get

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

This can be written as AB=C. Hence we get $B = A^{-1}C$. Reshaping the B vector gives the Homography matrix.

2 Steps

2.1 Task 1a

Project the kitten image given in Fig.1d onto the images in Fig. 1a, 1b and 1c.

1. Find the homography matrix H using the 4 given corner points of the two images.The domain image is 1d and the range image is 1a,1b and 1c.Use the method described in the Logic section.

2. Using the homography matrix H, project each point of the domain image (kitten) to the range image. (Equation 1)

3. The projected points are not integer values, hence we round off the points and convert to integer. Replace the pixel values of the range image with the corresponding pixel values of the domain image.

2.2 Task 1b

Find the homography between image 1a and 1b, and between 1b and 1c. Multiply the two homographies and apply the new homography matrix to 1a.

1. Find the homography matrix H1 using the 4 given corner points of the two images.The domain image is 1a and the range image is 1b.Use the method described in the Logic section. Perform the same operation on images 1b and 1c to get H2. Multiply the H1 and H2 to get H.

2. Using the homography matrix H, project each point of the domain image (1a) to the range image(1c). (Equation 1)

3. The projected points are not integer values, hence we round off the points and convert to integer. Replace the pixel values of the range image with the corresponding pixel values of the domain image.

3 Corner Points used

Range Images:

	Image 1a	Image 1b	Image 1c
P	[295,507]	[336,691]	[107,438]
Q	[236,1602]	[332,2330]	[119,1361]
R	[1680,1824]	[1881,1998]	[1093,1859]
S	[1771,359]	[1885,749]	[1212,301]

Domain Image:

	Image 1d
P	[0,0]
Q	[0,1125]
R	[1920,1125]
S	[1920,0]

4 Results

4.1 Input Images

Figure 1: 1a

Figure 2: 1b

Figure 3: 1c

Figure 4: 1d

4.2 Projected Images

Figure 5: 1d on 1a

Figure 6: 1d on 1b

Figure 7: 1d on 1c

Figure 8: Resultant Homography on 1a

5 Task 2

For task 2 we have to perform the same operations as task 1a and 1b on our own images.

5.1 Input Images

Figure 9: 2a

Figure 10: 2b

Figure 11: 2c

Figure 12: 2d

5.2 Corner Points

Range Images:

	Image 1a	Image 1b	Image 1c
P	[1939,753]	[1939,753]	[1860,925]
Q	[1890,2275]	[1890,2275]	[1491,1979]
R	[2717,2235]	[2717,2235]	[2505,2462]
S	[2702,1108]	[2702,1108]	[2929,1162]

Domain Image:

	Image 1d
P	[0,0]
Q	[0,1493]
R	[1376,1493]
S	[1376,0]

5.3 Projected Images

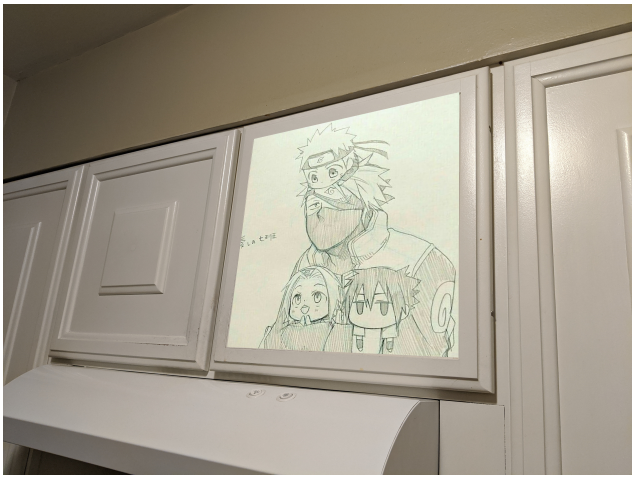

Figure 13: 2d on 2a

Figure 14: 2d on 2b

Figure 15: 2d on 2c

Figure 16: Resultant Homography on 2a

6 Note

Initially on trying out the method with considering kitten as the domain image and frame as the range image did not yield good results. This was because every point in the domain did not map to every point in the range. Hence some pixels in the range image did not have any corresponding points in the domain image.
Method: (Please refer to the Homography mapping function)

1. Get 4 points in the domain image X (kittens)
2. Get 4 points in the range image X'(frame)
3. find the Homography H using the aforementioned steps
4. Find the corresponding points of X in the range image. X'=HX
5. Replace the X' points of range image with pixel values of X points in the domain image.

Figure 17: Projected Image. We can clearly see some pixels of the original image (frame) still in the output

This method did not ensure each pixel in the range image was getting replaced. Here is the result of the projection.

Hence implemented another logic (Homographymapping2 function), where the points from the range image are mapped to the domain image. This method ensures that all the pixels in the range image have a corresponding pixel in the domain image. This leads to a clean projection of the domain image onto the range image.

7 Code Listing

```
# -*- coding: utf-8 -*-
"""hw2_Suyash_Ail.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1O7WF_2eHd89m1W1_xU8dbM9oPMcBW_Ys
"""

import numpy as np
import matplotlib.pyplot as plt
import cv2
from PIL import Image

#get the 4 points of the images 1a,1b and 1c. Also return a vector X_prime=[x1
    ,y1,x2,y2,x3,y3,x4,y4]
def get_4_points(n):
  if n==1:  #image 1a
    #for the image
    P=[295,507]    #x1,y1
    Q=[236,1602]   #x2,y2
```

```
    R=[1680,1824] #x3,y3
    S=[1771,359]  #x4,y4
    X_prime=P+Q+R+S
    return np.array([P,Q,R,S]),X_prime

  if n==2:    #image 1b
    #for the image
    P=[336,691]
    Q=[332,2330]
    R=[1881,1998]
    S=[1885,749]
    X_prime=P+Q+R+S
    return np.array([P,Q,R,S]),X_prime

  if n==3:    #image 1c
    #for the image
    P=[107,438]   #x1,y1
    Q=[119,1361]  #x2,y2
    R=[1093,1859] #x3,y3
    S=[1212,301]  #x4,y4
    X_prime=P+Q+R+S
    return np.array([P,Q,R,S]),X_prime

#get the 4 corner points of the cat image
def get_cat_points():
  #for the kittens
  P_dash=[0,0]
  Q_dash=[0,1125]
  R_dash=[1920,1125]
  S_dash=[1920,0]
  X=P_dash+Q_dash+R_dash+S_dash
  return np.array([P_dash,Q_dash,R_dash,S_dash]),X

#to check to validitiy of the 4 points, we draw a bounding box aroud the image
    .
def draw_bbox(image,points):
  copy_image=image.copy()
  #draw bounding box to check correctnes of the points
  points_reshape = points.reshape((-1, 1, 2))
  color = (255, 0, 0)
  im1_bbox = cv2.polylines(copy_image, [points_reshape], isClosed=True,color=
    color,thickness = 10)
  cv2.imwrite("bbox.jpg",im1_bbox)
  plt.axis('off')
  plt.imshow(im1_bbox)

def Homography_matrix(X_prime,X):
  '''
  This function takes input X and X_prime and computes the homography between
    them. Returns a 3x3 non-singular matrix
  H.
  '''
  A = np.zeros((8,8)) #initialise a matrix
  H = np.ones((3,3))

  C=X_prime

  #build the A matrix column by column
```

```
  for i in range(0,4):
    A[2*i,0]=X[2*i]
    A[2*i,1]=X[2*i+1]
    A[2*i,2]=1
    A[2*i+1,3]=X[2*i]
    A[2*i+1,4]=X[2*i+1]
    A[2*i+1,5]=1
    A[2*i,6]=X[2*i]*X_prime[2*i]
    A[2*i+1,6]=X[2*i]*X_prime[2*i+1]
    A[2*i,7]=(X[2*i+1]*X_prime[2*i])
    A[2*i+1,7]=(X[2*i+1]*X_prime[2*i+1])
  A[:,6:]=-A[:,6:]

  #B=(A^-1)C
  B=np.dot(np.linalg.inv(A),C)

  #reshape the B vector to get the homography matrix
  H[0,0]=B[0]
  H[0,1]=B[1]
  H[0,2]=B[2]
  H[1,0]=B[3]
  H[1,1]=B[4]
  H[1,2]=B[5]
  H[2,0]=B[6]
  H[2,1]=B[7]
  H[2,2]=1
  return H

def get_mask_image(image,image_points):
    '''
    This function returns a binary image of the passed image with all points outside the bounding box 0. This is done
    to make it easier to pick the pixels that are present inside the bounding box during projection.
    '''
    image_empty = np.zeros(image.shape,dtype='uint8')
    image_points_reshape=image_points.reshape((-1, 1, 2))
    image_filled = cv2.fillPoly(image_empty,[image_points_reshape],(255,255,255))
    plt.imshow(image_filled)
    return image_filled

def Homography_mapping(im1,cat,H,image_filled):
  '''
  This function maps each pixel in the domain to the corresponding pixel in the range using the equation: X'=HX where
  X, X' are the homogeneous coordinates of the pixels in the domain and range.
  '''
  for i in range(0,cat.shape[1]): #X-cordinate
    for j in range(0,cat.shape[0]): #Y-coordinate
            Image_homo = np.array((i,j,1))  #create the homogeneous vector
            Image_mapped = np.matmul(H, Image_homo)   #HX
            Image_mapped = Image_mapped/Image_mapped[2]   #divide by x3 to get world coordinates
            Image_mapped = Image_mapped.astype(int)     #convert floating values to integer
            #if(any(image_filled[Image_mapped[1],Image_mapped[0]]>0)):
            im1[Image_mapped[1],Image_mapped[0]] = cat[j,i]   #replace the
```

```
    range pixels with corresponding pixels from the domain

  plt.imshow(im1)
  im1=cv2.cvtColor(im1, cv2.COLOR_RGB2BGR)
  cv2.imwrite("output.jpg",im1)

def Homography_mapping2(im1,cat,H,image_filled):
  '''
  This function maps each pixel in the domain to the corresponding pixel in
    the range using the equation: X'=HX where
  X, X' are the homogeneous coordinates of the pixels in the domain and range.
    This function finds the projection of
  image 1a onto the cat image. Running Homography_mapping and
    Homography_mapping2 shows better results for Homography_mapping2
  as all the pixels of the range image has a corresponding pixel in the domain
    image.
  '''
  for i in range(0,im1.shape[1]):
    for j in range(0,im1.shape[0]):
      if any(image_filled[j][i]) > 0:
        #traverse only through the points inside the bbox. accomplised using
    mask image.
        Image_homo = np.array((i,j,1))
        Image_mapped = np.matmul(H, Image_homo)
        Image_mapped = np.rint(Image_mapped/Image_mapped[2])
        Image_mapped = Image_mapped.astype(int)
        if(Image_mapped[0] < cat.shape[1] and Image_mapped[1] < cat.shape[0]):
          #check if all projected pixels do not exceed the image boundary
          im1[j,i] = cat[Image_mapped[1],Image_mapped[0]]
  return im1

"""Task 1.a"""

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting1.jpeg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

plt.figure()
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(cat)

P1,X_prime=get_4_points(1)
P_cat,X=get_cat_points()
draw_bbox(im1,P1)
plt.figure()
draw_bbox(cat,P_cat)

H=Homography_matrix(X,X_prime)
print(H)

image_filled=get_mask_image(im1,P1)
projected_image=Homography_mapping2(im1,cat,H,image_filled)
plt.imshow(projected_image)
projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output1.jpg",projected_image)
```

```
"""Task 1.b"""

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting2.jpeg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

plt.figure()
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(cat)

P1,X_prime=get_4_points(2)
P_cat,X=get_cat_points()
draw_bbox(im1,P1)
plt.figure()
draw_bbox(cat,P_cat)

H=Homography_matrix(X,X_prime)
print(H)

image_filled=get_mask_image(im1,P1)
projected_image=Homography_mapping2(im1,cat,H,image_filled)
plt.imshow(projected_image)
projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output2.jpg",projected_image)

"""Task 1.c"""

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting3.jpeg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

plt.figure()
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(cat)

P1,X_prime=get_4_points(3)
P_cat,X=get_cat_points()
draw_bbox(im1,P1)
plt.figure()
draw_bbox(cat,P_cat)

H=Homography_matrix(X,X_prime)
print(H)

image_filled=get_mask_image(im1,P1)
projected_image=Homography_mapping2(im1,cat,H,image_filled)
plt.imshow(projected_image)
projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output3.jpg",projected_image)

"""Task 1.d"""
```

```
238
239 im2=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting2.jpeg")
240 im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
241
242 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting1.jpeg")
243 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
244
245 im3=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting3.jpeg")
246 im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2RGB)
247
248 P1,X1=get_4_points(1)
249 P2,X2=get_4_points(2)
250 P3,X3=get_4_points(3)
251
252 #H1=Homography_matrix(X1,X2)
253 #H2=Homography_matrix(X2,X3)
254 H1=Homography_matrix(X2,X1)
255 H2=Homography_matrix(X3,X2)
256 H=np.matmul(H2,H1)
257
258 blank_image=np.zeros(im3.shape,dtype='uint8')
259 for i in range(0,im1.shape[1]): #X-cordinate, col
260     for j in range(0,im1.shape[0]): #Y-coordinate, row
261             im1_homo=np.array((i,j,1))
262             mapped_image = np.matmul(H, im1_homo)
263             mapped_image = np.rint(mapped_image/mapped_image[-1]) #Normalize
    and round off to nearest integer
264             mapped_image = mapped_image.astype(int)
265             if(mapped_image[0] > 0 and mapped_image[1] > 0 and mapped_image[0]
    < im3.shape[1] and mapped_image[1] < im3.shape[0]):
266                 blank_image[mapped_image[1],mapped_image[0]] = im1[j,i]
267
268 plt.imshow(blank_image)
269 blank_image=cv2.cvtColor(blank_image, cv2.COLOR_RGB2BGR)
270 cv2.imwrite("output4.jpg",blank_image)
271
272 """Task 2
273
274 Perform the same operations as done in task 1 but on custom images of your own
    .
275 """
276
277 def get_4_points(n):
278   if n==1:
279     #for the image
280     P=[1609,866]   #x1,y1
281     Q=[1437,2201]  #x2,y2
282     R=[2919,2270] #x3,y3
283     S=[2924,566]  #x4,y4
284     X_prime=P+Q+R+S
285     return np.array([P,Q,R,S]),X_prime
286
287   if n==2:
288     #for the image
289     P=[1939,753]
290     Q=[1890,2275]
291     R=[2717,2235]
292     S=[2702,1108]
```

```
    X_prime=P+Q+R+S
    return np.array([P,Q,R,S]),X_prime

  if n==3:
    #for the image
    P=[1860,925]   #x1,y1
    Q=[1491,1979]  #x2,y2
    R=[2505,2462] #x3,y3
    S=[2929,1162]  #x4,y4
    X_prime=P+Q+R+S
    return np.array([P,Q,R,S]),X_prime

def get_cat_points():
  #for the kittens
  P_dash=[0,0]
  Q_dash=[0,1493]
  R_dash=[1376,1493]
  S_dash=[1376,0]
  X=P_dash+Q_dash+R_dash+S_dash
  return np.array([P_dash,Q_dash,R_dash,S_dash]),X

"""Task 2.a"""

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im1.jpg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

plt.figure()
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/naruto.png")
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(cat)

P1,X_prime=get_4_points(1)
P_cat,X=get_cat_points()
draw_bbox(im1,P1)
plt.figure()
draw_bbox(cat,P_cat)

H=Homography_matrix(X,X_prime)
print(H)

image_filled=get_mask_image(im1,P1)
projected_image=Homography_mapping2(im1,cat,H,image_filled)
plt.imshow(projected_image)
projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output21.jpg",projected_image)

"""Task 2.b"""

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im2.jpg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

plt.figure()
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/naruto.png")
```

```
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(cat)

P1,X_prime=get_4_points(2)
P_cat,X=get_cat_points()
draw_bbox(im1,P1)
plt.figure()
draw_bbox(cat,P_cat)

H=Homography_matrix(X,X_prime)
print(H)

image_filled=get_mask_image(im1,P1)
projected_image=Homography_mapping2(im1,cat,H,image_filled)
plt.imshow(projected_image)
projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output22.jpg",projected_image)

"""Task 2.c"""

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im3.jpg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

plt.figure()
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/naruto.png")
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(cat)

P1,X_prime=get_4_points(3)
P_cat,X=get_cat_points()
draw_bbox(im1,P1)
plt.figure()
draw_bbox(cat,P_cat)

H=Homography_matrix(X,X_prime)
print(H)

image_filled=get_mask_image(im1,P1)
projected_image=Homography_mapping2(im1,cat,H,image_filled)
plt.imshow(projected_image)
projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output23.jpg",projected_image)

"""Task 2.d"""

im2=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im2.jpg")
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im1.jpg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

im3=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im3.jpg")
im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2RGB)

```

```
P1,X1=get_4_points(1)
P2,X2=get_4_points(2)
P3,X3=get_4_points(3)

H1=Homography_matrix(X2,X1)
H2=Homography_matrix(X3,X2)
H=np.matmul(H2,H1)

blank_image=np.zeros(im3.shape,dtype='uint8')
for i in range(0,im1.shape[1]): #X-cordinate, col
    for j in range(0,im1.shape[0]): #Y-coordinate, row
            im1_homo=np.array((i,j,1))
            mapped_image = np.matmul(H, im1_homo)
            mapped_image = np.rint(mapped_image/mapped_image[-1]) #Normalize and round off to nearest integer
            mapped_image = mapped_image.astype(int)
            if(mapped_image[0] > 0 and mapped_image[1] > 0 and mapped_image[0] < im3.shape[1] and mapped_image[1] < im3.shape[0]):
                blank_image[mapped_image[1],mapped_image[0]] = im1[j,i]

plt.imshow(blank_image)
projected_image=cv2.cvtColor(blank_image, cv2.COLOR_RGB2BGR)
cv2.imwrite("output24.jpg",projected_image)
```

8 Code for trying out more than 4 corresponding points

Here we use more than 4 points for estimating the Homography. Midpoint of the line joining the 4 points have been considered as additional points. Hence we have 8 points. Since the matrix A is no longer a square matrix (16x8), we make use of pseudo-inverse (Python: np.linalg.pinv) to calculate A^{-1}. Rest of the procedure is the same as task 1.

Figure 18: Projection of 1d onto 1a using 8 points.

```
#task 1.a

im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting1.jpeg")
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
plt.imshow(im1)

cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
plt.imshow(cat)

#for the image
P=[295,507]    #x1,y1
Q=[236,1602]   #x2,y2
R=[1680,1824] #x3,y3
S=[1771,359]   #x4,y4
T= [(P[0]+Q[0])/2,(P[1]+Q[1])/2]     #x5,y5
U=[(Q[0]+R[0])/2,(Q[1]+R[1])/2]      #x6,y6
V=[(R[0]+S[0])/2,(R[1]+S[1])/2]      #x7,y7
W=[(S[0]+P[0])/2,(S[1]+P[1])/2]      #x8,y8

#for the kittens
P_dash=[0,0]          #x1',y1'
Q_dash=[0,1125]       #x2',y2'
R_dash=[1920,1125]    #x3',y3'
S_dash=[1920,0]       #x4',y4'
#extra points
```

```
T_dash=[(P_dash[0]+Q_dash[0])/2,(P_dash[1]+Q_dash[1])/2]     #x5',y5'
U_dash=[(Q_dash[0]+R_dash[0])/2,(Q_dash[1]+R_dash[1])/2]     #x6',y6'
V_dash=[(R_dash[0]+S_dash[0])/2,(R_dash[1]+S_dash[1])/2]     #x7',y7'
W_dash=[(S_dash[0]+P_dash[0])/2,(S_dash[1]+P_dash[1])/2]     #x8',y8'

#draw bounding box to check correctnes of the points
im1_points=np.array([P,Q,R,S])
im1_points_reshape = im1_points.reshape((-1, 1, 2))
color = (255, 0, 0)
im1_bbox = cv2.polylines(im1, [im1_points_reshape], isClosed=True,color=color,
    thickness = 5)
plt.axis('off')
plt.imshow(im1_bbox)

#draw bounding box to check correctnes of the points
kittens_points=np.array([P_dash,Q_dash,R_dash,S_dash])
kittens_points_reshape = kittens_points.reshape((-1, 1, 2))
color = (255, 0, 0)
kittens_bbox = cv2.polylines(cat, [kittens_points_reshape], isClosed=True,
    color=color,thickness = 5)
plt.axis('off')
plt.imshow(kittens_bbox)

X=P+Q+R+S+T+U+V+W    #build the X vector X=[x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,
    y6,x7,y7,x8,y8]
X_prime=P_dash+Q_dash+R_dash+S_dash+T_dash+U_dash+V_dash+W_dash   #build the X
    ' vector X'=[x1',y1',x2',y2',x3',y3',x4',y4',x'5,y'5,x'6,y'6,x'7,y'7,x'8,y
    '8]
X_prime

#initialize matrices for A and H
A = np.zeros((16,8))
H = np.ones((3,3))

C_prime=X_prime
C=X

#find the A matrix
for i in range(0,8):
  A[2*i,0]=C[2*i]
  A[2*i,1]=C[2*i+1]
  A[2*i,2]=1
  A[2*i+1,3]=C[2*i]
  A[2*i+1,4]=C[2*i+1]
  A[2*i+1,5]=1
  A[2*i,6]=C[2*i]*C_prime[2*i]
  A[2*i+1,6]=C[2*i]*C_prime[2*i+1]
  A[2*i,7]=(C[2*i+1]*C_prime[2*i])
  A[2*i+1,7]=(C[2*i+1]*C_prime[2*i+1])
A[:,6:]=-A[:,6:]
A

#B=(A^-1)C_prime
B=np.dot(np.linalg.pinv(A),C_prime)

#reshape the B vector to form the H matrix
H[0,0]=B[0]
H[0,1]=B[1]
```

```
H[0,2]=B[2]
H[1,0]=B[3]
H[1,1]=B[4]
H[1,2]=B[5]
H[2,0]=B[6]
H[2,1]=B[7]
H[2,2]=1
H

## project the kitten image onto the painting

#build a mask image so as to make sure the proejcted image(domain) doesn't exceed the range image
new_image = np.zeros(im1.shape,dtype='uint8')
image_filled = cv2.fillPoly(new_image,[im1_points_reshape],(255,255,255))
plt.imshow(image_filled)

#rows are the y-coordinates and columns are the x-coordinates
for i in range(im1.shape[1]): #X-cordinate
    for j in range(im1.shape[0]): #Y-coordinate
        if any(image_filled[j][i]) > 0:     #check only for pixels within the mask image
            X_domain=np.array((i,j,1))      #get the homogenous coordinate of frame
            X_range = np.matmul(H, X_domain)      #perform homography
            X_range = np.rint(X_range/X_range[2]) #divide by the third element and round-off the value
            X_range = X_range.astype(int)       #convert the matrix to int
            #check if the mapped pixels lie within the cat's image size
            if(X_range[0] < cat.shape[1] and X_range[1] < cat.shape[0]):
                #replace the frame pixels with corresponding cat pixels
                im1[j][i] = cat[X_range[1]][X_range[0]]

plt.imshow(im1)
cv2.imwrite("task1.jpg",im1)
```