

# 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  $3 \times 3$  matrix  $H$  which is called the homography matrix.

$$\vec{x}' = H\vec{x} \quad (1)$$

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

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

The matrix  $H$  is

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

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

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

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

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

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

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

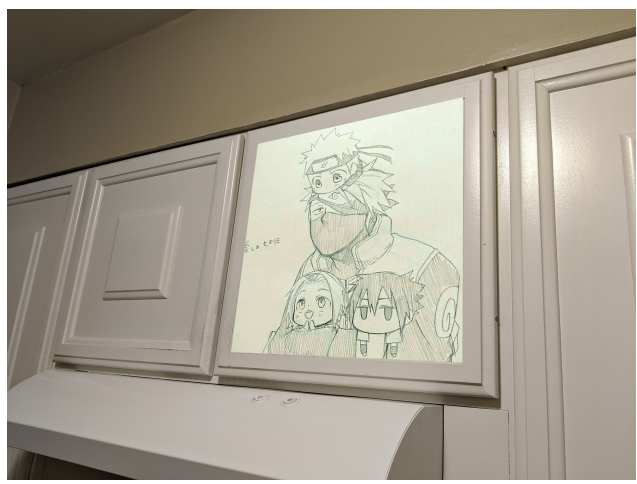


Figure 13: 2d on 2a



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

```

1 #-*- coding: utf-8 -*-
2 """hw2_Suyash_Ail.ipynb
3
4 Automatically generated by Colaboratory.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1O7WF_2eHd89m1W1_xU8dbM9oPMcBW_Ys
8 """
9
10 import numpy as np
11 import matplotlib.pyplot as plt
12 import cv2
13 from PIL import Image
14
15 #get the 4 points of the images 1a,1b and 1c. Also return a vector X_prime=[x1
16     ,y1,x2,y2,x3,y3,x4,y4]
17 def get_4_points(n):
18     if n==1: #image 1a
19         #for the image
20         P=[295,507] #x1,y1
21         Q=[236,1602] #x2,y2

```

```

21     R=[1680,1824] #x3 , y3
22     S=[1771,359] #x4 , y4
23     X_prime=P+Q+R+S
24     return np.array ([P,Q,R,S] ) ,X_prime
25
26     if n==2: #image 1b
27         #for the image
28         P=[336,691]
29         Q=[332,2330]
30         R=[1881,1998]
31         S=[1885,749]
32         X_prime=P+Q+R+S
33         return np.array ([P,Q,R,S] ) ,X_prime
34
35     if n==3: #image 1c
36         #for the image
37         P=[107,438] #x1 ,y1
38         Q=[119,1361] #x2 ,y2
39         R=[1093,1859] #x3 ,y3
40         S=[1212,301] #x4 ,y4
41         X_prime=P+Q+R+S
42         return np.array ([P,Q,R,S] ) ,X_prime
43
44 #get the 4 corner points of the cat image
45 def get_cat_points () :
46     #for the kittens
47     P_dash=[0,0]
48     Q_dash=[0,1125]
49     R_dash=[1920,1125]
50     S_dash=[1920,0]
51     X=P_dash+Q_dash+R_dash+S_dash
52     return np.array ([P_dash,Q_dash,R_dash,S_dash] ) ,X
53
54 #to check to validity of the 4 points , we draw a bounding box around the image
55
56 def draw_bbox(image , points ) :
57     copy_image=image.copy ()
58     #draw bounding box to check correctnes of the points
59     points_reshape = points.reshape((-1, 1, 2))
60     color = (255, 0, 0)
61     im1_bbox = cv2.polylines(copy_image , [points_reshape] , isClosed=True , color=
62         color , thickness = 10)
63     cv2.imwrite("bbox.jpg" ,im1_bbox)
64     plt.axis('off')
65     plt.imshow(im1_bbox)
66
67 def Homography_matrix(X_prime,X) :
68     '''
69     This function takes input X and X_prime and computes the homography between
70     them. Returns a 3x3 non-singular matrix
71     H.
72     '''
73     A = np.zeros((8,8)) #initialise a matrix
74     H = np.ones((3,3))
75     C=X_prime
76     #build the A matrix column by column

```

```

76 for i in range(0,4):
77     A[2*i,0]=X[2*i]
78     A[2*i,1]=X[2*i+1]
79     A[2*i,2]=1
80     A[2*i+1,3]=X[2*i]
81     A[2*i+1,4]=X[2*i+1]
82     A[2*i+1,5]=1
83     A[2*i,6]=X[2*i]*X_prime[2*i]
84     A[2*i+1,6]=X[2*i]*X_prime[2*i+1]
85     A[2*i,7]=(X[2*i+1]*X_prime[2*i])
86     A[2*i+1,7]=(X[2*i+1]*X_prime[2*i+1])
87 A[:,6:]=-A[:,6:]
88
89 #B=(A^-1)C
90 B=np.dot(np.linalg.inv(A),C)
91
92 #reshape the B vector to get the homography matrix
93 H[0,0]=B[0]
94 H[0,1]=B[1]
95 H[0,2]=B[2]
96 H[1,0]=B[3]
97 H[1,1]=B[4]
98 H[1,2]=B[5]
99 H[2,0]=B[6]
100 H[2,1]=B[7]
101 H[2,2]=1
102 return H
103
104 def get_mask_image(image, image_points):
105     '''
106     This function returns a binary image of the passed image with all points
107     outside the bounding box 0. This is done
108     to make it easier to pick the pixels that are present inside the bounding
109     box during projection.
110     '''
111     image_empty = np.zeros(image.shape, dtype='uint8')
112     image_points_reshape=image_points.reshape((-1, 1, 2))
113     image_filled = cv2.fillPoly(image_empty, [image_points_reshape
114     ],(255,255,255))
115     plt.imshow(image_filled)
116     return image_filled
117
118 def Homography_mapping(im1, cat, H, image_filled):
119     '''
120     This function maps each pixel in the domain to the corresponding pixel in
121     the range using the equation: X'=HX where
122     X, X' are the homogeneous coordinates of the pixels in the domain and range.
123     '''
124     for i in range(0,cat.shape[1]): #X- cordinate
125         for j in range(0,cat.shape[0]): #Y- cordinate
126             Image_homo = np.array((i, j, 1)) #create the homogeneous vector
127             Image_mapped = np.matmul(H, Image_homo) #HX
128             Image_mapped = Image_mapped/Image_mapped[2] #divide by x3 to get
129             world coordinates
130             Image_mapped = Image_mapped.astype(int) #convert floating
131             values to integer
132             #if (any(image_filled [Image_mapped[1],Image_mapped[0]]>0)):
133             im1 [Image_mapped[1],Image_mapped[0]] = cat [j, i] #replace the

```

```

    range pixels with corresponding pixels from the domain
128
129 plt.imshow(im1)
130 im1=cv2.cvtColor(im1, cv2.COLOR_RGB2BGR)
131 cv2.imwrite("output.jpg",im1)
132
133 def Homography_mapping2(im1,cat,H,image_filled):
134     '''
135     This function maps each pixel in the domain to the corresponding pixel in
136     the range using the equation:  $X'=HX$  where
137     X, X' are the homogeneous coordinates of the pixels in the domain and range.
138     This function finds the projection of
139     image 1a onto the cat image. Running Homography_mapping and
140     Homography_mapping2 shows better results for Homography_mapping2
141     as all the pixels of the range image has a corresponding pixel in the domain
142     image.
143     '''
144     for i in range(0,im1.shape[1]):
145         for j in range(0,im1.shape[0]):
146             if any(image_filled[j][i]) > 0:
147                 #traverse only through the points inside the bbox. accomplished using
148                 mask image.
149                 Image_homo = np.array((i,j,1))
150                 Image_mapped = np.matmul(H, Image_homo)
151                 Image_mapped = np rint (Image_mapped/Image_mapped[2])
152                 Image_mapped = Image_mapped.astype(int)
153                 if (Image_mapped[0] < cat.shape[1] and Image_mapped[1] < cat.shape[0]):
154                     #check if all projected pixels do not exceed the image boundary
155                     im1[j,i] = cat [Image_mapped[1],Image_mapped[0]]
156     return im1
157
158 """Task 1.a"""
159
160 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting1.jpeg")
161 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
162
163 plt.figure()
164 plt.imshow(im1)
165
166 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
167 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
168 plt.figure()
169 plt.imshow(cat)
170
171 P1,X_prime=get_4_points(1)
172 P_cat,X=get_cat_points()
173 draw_bbox(im1,P1)
174 plt.figure()
175 draw_bbox(cat,P_cat)
176
177 H=Homography_matrix(X,X_prime)
178 print(H)
179
180 image_filled=get_mask_image(im1,P1)
181 projected_image=Homography_mapping2(im1,cat,H,image_filled)
182 plt.imshow(projected_image)
183 projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
184 cv2.imwrite("output1.jpg",projected_image)

```

```

180
181 """Task 1.b"""
182
183 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting2.jpeg")
184 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
185
186 plt.figure()
187 plt.imshow(im1)
188
189 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
190 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
191 plt.figure()
192 plt.imshow(cat)
193
194 P1,X_prime=get_4_points(2)
195 P_cat,X=get_cat_points()
196 draw_bbox(im1,P1)
197 plt.figure()
198 draw_bbox(cat,P_cat)
199
200 H=Homography_matrix(X,X_prime)
201 print(H)
202
203 image_filled=get_mask_image(im1,P1)
204 projected_image=Homography_mapping2(im1,cat,H,image_filled)
205 plt.imshow(projected_image)
206 projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
207 cv2.imwrite("output2.jpg",projected_image)
208
209 """Task 1.c"""
210
211 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting3.jpeg")
212 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
213
214 plt.figure()
215 plt.imshow(im1)
216
217 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
218 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
219 plt.figure()
220 plt.imshow(cat)
221
222 P1,X_prime=get_4_points(3)
223 P_cat,X=get_cat_points()
224 draw_bbox(im1,P1)
225 plt.figure()
226 draw_bbox(cat,P_cat)
227
228 H=Homography_matrix(X,X_prime)
229 print(H)
230
231 image_filled=get_mask_image(im1,P1)
232 projected_image=Homography_mapping2(im1,cat,H,image_filled)
233 plt.imshow(projected_image)
234 projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
235 cv2.imwrite("output3.jpg",projected_image)
236
237 """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-coordinate, 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]

```

```

293     X_prime=P+Q+R+S
294     return np.array ([P,Q,R,S] ),X_prime
295
296     if n==3:
297         #for the image
298         P=[1860,925]     #x1 , y1
299         Q=[1491,1979]   #x2 , y2
300         R=[2505,2462]  #x3 , y3
301         S=[2929,1162]  #x4 , y4
302         X_prime=P+Q+R+S
303         return np.array ([P,Q,R,S] ),X_prime
304
305 def get_cat_points () :
306     #for the kittens
307     P_dash=[0,0]
308     Q_dash=[0,1493]
309     R_dash=[1376,1493]
310     S_dash=[1376,0]
311     X=P_dash+Q_dash+R_dash+S_dash
312     return np.array ([P_dash,Q_dash,R_dash,S_dash] ),X
313
314 """Task 2.a"""
315
316 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im1.jpg")
317 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
318
319 plt.figure()
320 plt.imshow(im1)
321
322 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/naruto.png")
323 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
324 plt.figure()
325 plt.imshow(cat)
326
327 P1,X_prime=get_4_points(1)
328 P_cat,X=get_cat_points()
329 draw_bbox(im1,P1)
330 plt.figure()
331 draw_bbox(cat,P_cat)
332
333 H=Homography_matrix(X,X_prime)
334 print(H)
335
336 image_filled=get_mask_image(im1,P1)
337 projected_image=Homography_mapping2(im1,cat,H,image_filled)
338 plt.imshow(projected_image)
339 projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
340 cv2.imwrite("output21.jpg",projected_image)
341
342 """Task 2.b"""
343
344 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im2.jpg")
345 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
346
347 plt.figure()
348 plt.imshow(im1)
349
350 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/naruto.png")

```

```

351 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
352 plt.figure()
353 plt.imshow(cat)
354
355 P1,X_prime=get_4_points(2)
356 P_cat,X=get_cat_points()
357 draw_bbox(im1,P1)
358 plt.figure()
359 draw_bbox(cat,P_cat)
360
361 H=Homography_matrix(X,X_prime)
362 print(H)
363
364 image_filled=get_mask_image(im1,P1)
365 projected_image=Homography_mapping2(im1,cat,H,image_filled)
366 plt.imshow(projected_image)
367 projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
368 cv2.imwrite("output22.jpg",projected_image)
369
370 """Task 2.c"""
371
372 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im3.jpg")
373 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
374
375 plt.figure()
376 plt.imshow(im1)
377
378 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/naruto.png")
379 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
380 plt.figure()
381 plt.imshow(cat)
382
383 P1,X_prime=get_4_points(3)
384 P_cat,X=get_cat_points()
385 draw_bbox(im1,P1)
386 plt.figure()
387 draw_bbox(cat,P_cat)
388
389 H=Homography_matrix(X,X_prime)
390 print(H)
391
392 image_filled=get_mask_image(im1,P1)
393 projected_image=Homography_mapping2(im1,cat,H,image_filled)
394 plt.imshow(projected_image)
395 projected_image=cv2.cvtColor(projected_image, cv2.COLOR_RGB2BGR)
396 cv2.imwrite("output23.jpg",projected_image)
397
398 """Task 2.d"""
399
400 im2=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im2.jpg")
401 im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
402
403 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im1.jpg")
404 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
405
406 im3=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/im3.jpg")
407 im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2RGB)
408

```

```

409 P1,X1=get_4_points(1)
410 P2,X2=get_4_points(2)
411 P3,X3=get_4_points(3)
412
413 H1=Homography_matrix(X2,X1)
414 H2=Homography_matrix(X3,X2)
415 H=np.matmul(H2,H1)
416
417 blank_image=np.zeros(im3.shape,dtype='uint8')
418 for i in range(0,im1.shape[1]): #X-coordinate, col
419     for j in range(0,im1.shape[0]): #Y-coordinate, row
420         im1_homo=np.array((i,j,1))
421         mapped_image = np.matmul(H, im1_homo)
422         mapped_image = np rint(mapped_image/mapped_image[-1]) #Normalize
         and round off to nearest integer
423         mapped_image = mapped_image.astype(int)
424         if(mapped_image[0] > 0 and mapped_image[1] > 0 and mapped_image[0]
         < im3.shape[1] and mapped_image[1] < im3.shape[0]):
425             blank_image[mapped_image[1],mapped_image[0]] = im1[j,i]
426
427 plt.imshow(blank_image)
428 projected_image=cv2.cvtColor(blank_image, cv2.COLOR_RGB2BGR)
429 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 ( $16 \times 8$ ), 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.

```
1 #task 1.a
2
3 im1=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/painting1.jpeg")
4 im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
5 plt.imshow(im1)
6
7 cat=cv2.imread("/content/drive/My Drive/hw2_Task1_Images/kittens.jpeg")
8 cat = cv2.cvtColor(cat, cv2.COLOR_BGR2RGB)
9 plt.imshow(cat)
10
11 #for the image
12 P=[295,507] #x1,y1
13 Q=[236,1602] #x2,y2
14 R=[1680,1824] #x3,y3
15 S=[1771,359] #x4,y4
16 T=[(P[0]+Q[0])/2,(P[1]+Q[1])/2] #x5,y5
17 U=[(Q[0]+R[0])/2,(Q[1]+R[1])/2] #x6,y6
18 V=[(R[0]+S[0])/2,(R[1]+S[1])/2] #x7,y7
19 W=[(S[0]+P[0])/2,(S[1]+P[1])/2] #x8,y8
20
21 #for the kittens
22 P_dash=[0,0] #x1',y1'
23 Q_dash=[0,1125] #x2',y2'
24 R_dash=[1920,1125] #x3',y3'
25 S_dash=[1920,0] #x4',y4'
26 #extra points
```

```

27 T_dash=[(P_dash[0]+Q_dash[0])/2,(P_dash[1]+Q_dash[1])/2] #x5',y5'
28 U_dash=[(Q_dash[0]+R_dash[0])/2,(Q_dash[1]+R_dash[1])/2] #x6',y6'
29 V_dash=[(R_dash[0]+S_dash[0])/2,(R_dash[1]+S_dash[1])/2] #x7',y7'
30 W_dash=[(S_dash[0]+P_dash[0])/2,(S_dash[1]+P_dash[1])/2] #x8',y8'
31
32 #draw bounding box to check correctnes of the points
33 im1_points=np.array([P,Q,R,S])
34 im1_points_reshape = im1_points.reshape((-1, 1, 2))
35 color = (255, 0, 0)
36 im1_bbox = cv2.polylines(im1, [im1_points_reshape], isClosed=True,color=color,
    thickness = 5)
37 plt.axis('off')
38 plt.imshow(im1_bbox)
39
40 #draw bounding box to check correctnes of the points
41 kittens_points=np.array([P_dash,Q_dash,R_dash,S_dash])
42 kittens_points_reshape = kittens_points.reshape((-1, 1, 2))
43 color = (255, 0, 0)
44 kittens_bbox = cv2.polylines(cat, [kittens_points_reshape], isClosed=True,
    color=color,thickness = 5)
45 plt.axis('off')
46 plt.imshow(kittens_bbox)
47
48 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]
49 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]
50 X_prime
51
52 #initialize matrices for A and H
53 A = np.zeros((16,8))
54 H = np.ones((3,3))
55
56 C_prime=X_prime
57 C=X
58
59 #find the A matrix
60 for i in range(0,8):
61     A[2*i,0]=C[2*i]
62     A[2*i,1]=C[2*i+1]
63     A[2*i,2]=1
64     A[2*i+1,3]=C[2*i]
65     A[2*i+1,4]=C[2*i+1]
66     A[2*i+1,5]=1
67     A[2*i,6]=C[2*i]*C_prime[2*i]
68     A[2*i+1,6]=C[2*i]*C_prime[2*i+1]
69     A[2*i,7]=(C[2*i+1]*C_prime[2*i])
70     A[2*i+1,7]=(C[2*i+1]*C_prime[2*i+1])
71 A[:,6:]=-A[:,6:]
72 A
73
74 #B=(A^-1)C_prime
75 B=np.dot(np.linalg.pinv(A),C_prime)
76
77 #reshape the B vector to form the H matrix
78 H[0,0]=B[0]
79 H[0,1]=B[1]

```

```

80 H[0,2]=B[2]
81 H[1,0]=B[3]
82 H[1,1]=B[4]
83 H[1,2]=B[5]
84 H[2,0]=B[6]
85 H[2,1]=B[7]
86 H[2,2]=1
87 H
88
89 ## project the kitten image onto the painting
90
91 #build a mask image so as to make sure the proejected image(domain) doesn't
    exceed the range image
92 new_image = np.zeros(im1.shape, dtype='uint8')
93 image_filled = cv2.fillPoly(new_image, [im1_points_reshape], (255, 255, 255))
94 plt.imshow(image_filled)
95
96 #rows are the y-coordinates and columns are the x-coordinates
97 for i in range(im1.shape[1]): #X-cordinate
98     for j in range(im1.shape[0]): #Y-cordinate
99         if any(image_filled[j][i]) > 0: #check only for pixels within the
    mask image
100             X_domain=np.array((i, j, 1)) #get the homogenous coordinate of
    frame
101             X_range = np.matmul(H, X_domain) #perform homography
102             X_range = np rint(X_range/X_range[2]) #divide by the third element
    and round-off the value
103             X_range = X_range.astype(int) #convert the matrix to int
104             #check if the mapped pixels lie within the cat's image size
105             if(X_range[0] < cat.shape[1] and X_range[1] < cat.shape[0]):
106                 #replace the frame pixels with corresponding cat pixels
107                 im1[j][i] = cat[X_range[1]][X_range[0]]
108
109 plt.imshow(im1)
110 cv2.imwrite("task1.jpg", im1)

```