Department of Computer Engineering

Program:	Computer Engineering	Course:	CMPE-443 IPCV
Examination:	CEP	Session	2020
Maximum Marks:	30	Semester:	7th
Time allowed:	Till end of semester	Date:	16-11-2023

Instructions

- You are to work in a team of two to three peers to do this project.
- Only one solution to each problem must be given.
- All necessary working must be clearly shown to receive full credit.
- Solve the written portion of this project on this question paper. Use A4 white paper only, if extra pages
- are needed.

Sr. No	Description	Marks
1	CLO: 3 · Domain: Cognitive/5 · PLO: 4 One of the applications of computer vision is to enable a computer to be able to recognize traffic signs. In this project you will be implementing a portion of this application. A template for the sign representing disabled people has been provided to you in the image handicapped.png. The image captured.png is a captured imaged which contains the sign representing disabled people. *Manually* find out the coordinates of the four vertices bounding the sign, in both, the template as well as the captured image. These are going to be your 'interesting points'. (a) Which technique can you use to get a computer to match the 4 pairs of corresponding interesting points in the template and the captured image? (b) Explain how you can find out the homography between the template and the captured image using the corresponding points in part (a). (c) Explain how you can use this homography to warp the the sign present in the captured image so that it matches the dimensions of that present in the template. Marks = [4 + 8 + 8]	20
2	CLO: 3 · Domain: Cognitive/5 · PLO: 4 Implement your algorithms in Problem 1 parts (a) to (c) and show the result after warping the sign present in captured.png so that it matches the dimensions of that present handicapped.png.	10

***Manually* find out the coordinates of the four vertices bounding the sign, in both, the template as well as the captured image. These are going to be your 'interesting points'.**

Figure 1-Handicapped

Figure 2-Captured

Vertices from Handicapped image

i. V1(271,88)
ii. V2(320,153)
iii. V3(278,283)
iv. V4(372,263)

Vertices from Captured image

i. V11(337,118)
ii. V22(353,158)
iii. V33(215,213)
iv. V44(352,231)

a) Which technique can you use to get a computer to match the 4 pairs of corresponding interesting points in the template and the captured image?

Homography is a mathematical transformation that efficiently maps the points from one plane (in this case, the template image) to another plane (the captured image) in a way that preserves straight lines.

Homography in Matching Corresponding Points:

- **Identifying Corresponding Points:**
  First, we need to identify four pairs of corresponding points between the template and the captured image. These points can be features like corners, edges, or any distinct visual markers that are easily identifiable in both images.
- **Using Homography for Point Matching:**
  Once we have these point pairs, we can use them to compute a homography matrix. This matrix is a 3x3 transformation matrix that relates the coordinates of the points in the template image to their corresponding coordinates in the captured image.
- **Computing the Homography Matrix:**
  The homography matrix is computed using the coordinates of the corresponding points. This involves setting up a system of linear equations and solving them, typically using methods like **least squares** or **singular value decomposition (SVD).**
- **Applying the Homography:**
  After computing the homography matrix, we can apply it to the entire template image or any of its points. This transformation will map the template image onto the captured image, aligning the identified corresponding points.

b) Explain how you can find out the homography between the template and the captured image using the corresponding points in part (a).

Least Squares Technique for Point Matching:

The least squares technique is a mathematical method used to approximate the solution of an over-determined system (more equations than unknowns). In the context of matching points between images, it finds the best-fit transformation that minimizes the sum of the squares of the differences (errors) between the transformed points in one image and their corresponding points in another image.

- **Problem Setup**

  Two sets of corresponding points:

  Points from a template image: (x1,y1),(x2,y2),…,(xn,yn)

  Corresponding points from a captured image: x1′,y1′),(x2′,y2′),…,(xn′,yn′)

- **Homography Matrix**

  For a 2D homography, H is a 3x3 matrix:

  H =

$$
\begin{matrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & h33 \end{matrix}
$$

- **Setting Up Equations:**

  Each pair of corresponding points contributes to a set of equations. For instance, a homography requires two equations per point pair.

  The system of equations can be represented in matrix form as Ax=b, where A is a matrix constructed from the coordinates of the source points, x is a vector representing the flattened transformation matrix, and b is a vector constructed from the coordinates of the destination points.

$$
\begin{bmatrix}
271 & 88 & 1 & 0 & 0 & 0 & -(337)(271) & -(337)(88) & -337 \\
0 & 0 & 0 & 271 & 88 & 1 & -(118)(271) & -(118)(88) & -118 \\
320 & 153 & 1 & 0 & 0 & 0 & -(353)(320) & -(353)(153) & -353 \\
0 & 0 & 0 & 320 & 153 & 1 & -(158)(320) & -(158)(153) & -158 \\
278 & 283 & 1 & 0 & 0 & 0 & -(215)(278) & -(215)(283) & -215 \\
0 & 0 & 0 & 278 & 283 & 1 & (213)(278) & -(213)(283) & -213 \\
372 & 263 & 1 & 0 & 0 & 0 & -(352)(372) & -(352)(263) & -352 \\
0 & 0 & 0 & 372 & 263 & 1 & -(231)(372) & -(231)(263) & -231
\end{bmatrix}
\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
$$

$$Ah = 0.$$

- **Solving Using Least Squares:**

  To find the optimized solution of our problem we will use the technique called Least Squares. Mathematically, this is often solved using the normal equation

$$(ATA)x=ATb$$

The least squares solution aims to minimize the sum of the squares of the residuals (the differences between the observed values in b and the values predicted by our model Ax).

$$X=(A^tA)^{-1}A^Tb$$

- **Interpreting the Result:**

  The solution vector x gives the elements of the transformation matrix.

  This matrix can then be used to map any point from the template image to the captured image.

1.33806491e+00	-1.12399522e+00	1.07980718e+02
4.88357365e-01	1.37451407e-01	-1.42957688e+01
9.74024105e-04	-1.82998286e-0.3	1.00000000e+00

c) Explain how you can use this homography to warp the the sign present in the captured image so that it matches the dimensions of that present in the template.

- **Homography Matrix:**

  The homography matrix H is a 3x3 matrix defined as:

  H =

$$\begin{matrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & h33 \end{matrix}$$

- **Transforming Points:**

  Each point in the source image is represented in homogeneous coordinates. Now that we have the H matrix filled so we apply the homography matrix to this point, we get its new location in the destination image. The transformation is given by:

$$\mathbf{(xnew,ynew)=(\ x/\ w',\ y'/\ w')}$$

- **Applying to the Entire Image:**

  This transformation is applied to every pixel in the source image. The result is a new image where the perspective has been changed according to the homography. This is what "warping" refers to. We are applying forward wrap(i.e that we will apply the computed H matrix on the image “handicapped” and get the desire image “captured”)

- **Handling the Output Image Size:**

  The size of the output image (destination image) is choosen on the basis of the image on which we have to map our source image which is “handicapped”.

  The key is that the homography will map the points from the source image “Handicapped” to their corresponding locations in the output image space “Captured”.

Implementation:

Least squares:

```
import numpy as np

def compute_homography_manual_least_squares(src_points, dst_points):
    if len(src_points) < 4 or len(dst_points) < 4:
        raise ValueError("At least 4 point correspondences are required.")

    A = []
    b = []
    for (x, y), (x_prime, y_prime) in zip(src_points, dst_points):
        A.append([-x, -y, -1, 0, 0, 0, x*x_prime, y*x_prime])
        A.append([0, 0, 0, -x, -y, -1, x*y_prime, y*y_prime])
        b.extend([x_prime, y_prime])

    A = np.array(A)
    b = np.array(b)

    # Solve using the normal equation: (A^TA)x = A^Tb
    ATA = A.T.dot(A)
    ATb = A.T.dot(b)
    H = np.linalg.inv(ATA).dot(ATb)
    H = np.append(H, 1).reshape(3, 3)

    return H

# Provided points
src_points = np.float32([[271, 88], [320, 153], [278, 283], [372, 263]])
dst_points = np.float32([[337, 118], [353, 158], [215, 213], [352, 231]])

# Compute the homography matrix manually using least squares
H_manual_least_squares = compute_homography_manual_least_squares(src_points,
dst_points)
H_manual_least_squares

#Warping Image
import cv2
from google.colab.patches import cv2_imshow

# Load your image
image = cv2.imread('/content/handicapped.png')  # Replace with the path to
your image

# Homography matrix as you've calculated
H = np.array([[1.33806491e+00, -1.12399522e+00, 1.07980718e+02],
              [4.88357365e-01, 1.37451407e-01, -1.42957688e+01],
              [9.74024105e-04, -1.82998286e-03, 1.00000000e+00]])

# Determine the size of the output image
```

```
output_size = (image.shape[1], image.shape[0]) # Width and height of the original image

# Apply the warp perspective transformation
warped_image = cv2.warpPerspective(image, H, output_size)

# Save or display the warped image
cv2_imshow(warped_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# To save the image, you can use:
cv2.imwrite('/content/warped_image.jpg', warped_image)
```

Output H matrix:

```
array([[-1.33806491e+00,  1.12399522e+00, -1.07980718e+02],
       [-4.88357365e-01, -1.37451407e-01,  1.42957688e+01],
       [-9.74024105e-04,  1.82998286e-03,  1.00000000e+00]])
```

Warped Image: