Name : Aum M. Dhabalia **Date :** 02/09/2024

Roll No. : 21BEC027

Experiment…3 – Compute Homography Matrix

Objective :

- Estimate homography matrix without Singular Value Decomposition
- Implement Direct Learning Transformation (DLT) algorithm to estimate the homography matrix.
- Remove projective distortion in the image using DLT.

Import necessary libraries…

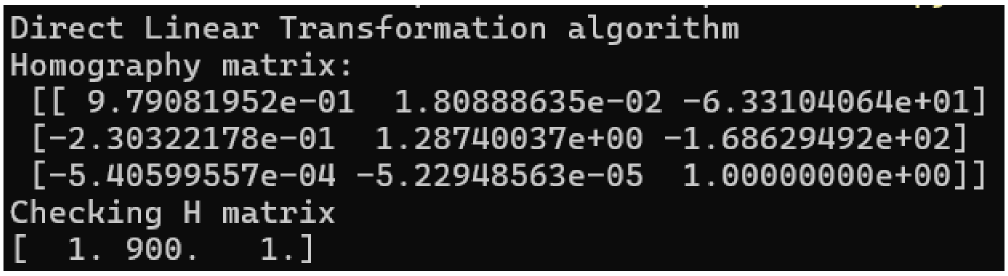
```
'''
Created on 2 September 2024 Mon 2:15:22 pm...
'''
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Task 1. Compute homography matrix for 4 given points without SVD and transform the point using computed homography matrix.

```
#Task 1
print("Direct Linear Transformation algorithm")
srcpt = np.array([[51,791],[63,143],[444,211],[426,719]])
dstpt = np.array([[1,900],[1,1],[501,1],[501,900]])
pt = np.array([51,791,1])
A = []
b = []
for i in range(4):
    x, y = srcpt[i][0],srcpt[i][1]
    x_prime,y_prime = dstpt[i][0],dstpt[i][1]
    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.append(x_prime)
    b.append(y_prime)
A = np.array(A)
b = np.array(b)
```

```
h = np.linalg.solve(A,b)

H = np.append(h,1).reshape(3,3)

print("Homography matrix:\n",H)

print("Checking H matrix")

check = np.matmul(H,pt)

check = check/check[2]

print(check)
```

Output



```
Direct Linear Transformation algorithm
Homography matrix:
 [[ 9.79081952e-01  1.80888635e-02 -6.33104064e+01]
 [-2.30322178e-01  1.28740037e+00 -1.68629492e+02]
 [-5.40599557e-04 -5.22948563e-05  1.00000000e+00]]
Checking H matrix
[  1. 900.   1.]
```

Observation :

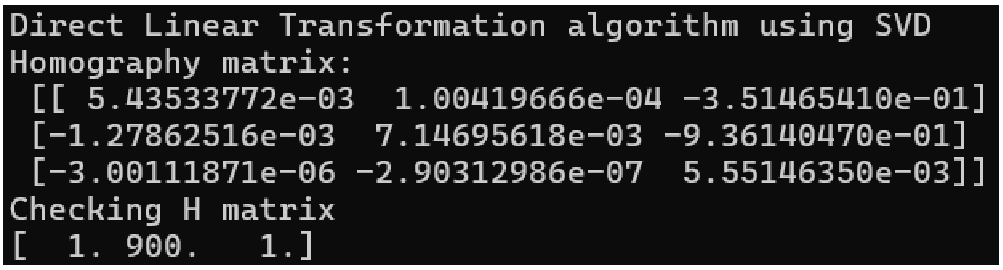
- The homography matrix is calculated using DLT algorithm.
- Here, H is divided by h_{33} element so that using minimum 4 source points and destination points, all other 8 elements of H is calculated.
- The matrix is verified by inserting the source point to obtain the same destination point.

Task 2. Compute homography matrix for 4 given points using DLT and transform the point using computed homography matrix.

```
#Task 2...

print("Direct Linear Transformation algorithm using SVD")

def homography_svd(pts_src, pts_dst):

  A = []

  for i in range(4):

    x, y = pts_src[i][0], pts_src[i][1]

    x_prime, y_prime = pts_dst[i][0], pts_dst[i][1]

    A.append([x,y,1,0,0,0,-x * x_prime,-y * x_prime,-x_prime])

    A.append([0,0,0,x,y,1,-x * y_prime,-y * y_prime,-y_prime])

  A = np.array(A)

  U, S, Vh = np.linalg.svd(A)

  H = Vh[-1].reshape(3,3)

  return H
```

```
H = homography_svd(srcpt,dstpt)

print("Homography matrix:\n",H)

print("Checking H matrix")

check = H@(pt)

check = check/check[2]

print(check)
```

Output



```
Direct Linear Transformation algorithm using SVD
Homography matrix:
 [[ 5.43533772e-03  1.00419666e-04 -3.51465410e-01]
 [-1.27862516e-03  7.14695618e-03 -9.36140470e-01]
 [-3.00111871e-06 -2.90312986e-07  5.55146350e-03]]
Checking H matrix
[  1. 900.   1.]
```

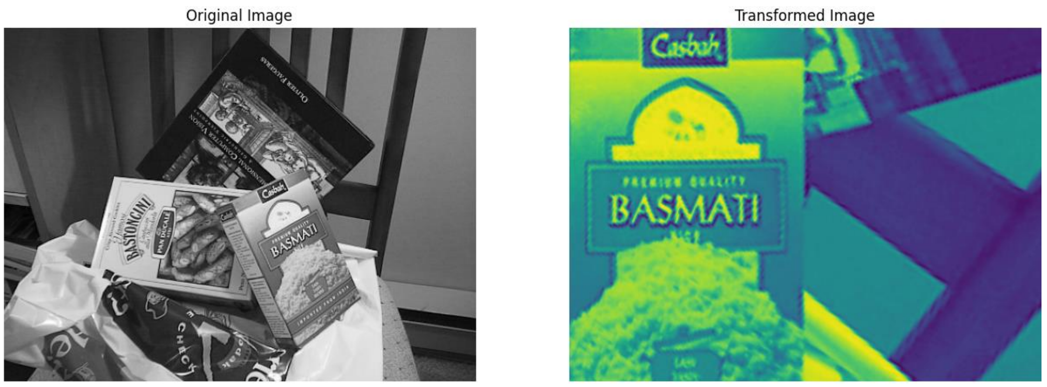
Observation :

- The homography matrix is calculated using DLT algorithm with SVD.
- The matrix is verified by inserting the source point to obtain the same destination point.

Task 3. A source image will be transformed into desired perspective view by computing the homography using DLT that maps the source points into the desired points.

```
I = cv2.imread(r"D:\Nirma Files\Computer Vision\box_sample.png")

srcpt = np.array([[247,190],[325,154],[319,347],[386,293]])

dstpt = np.array([[0,0],[250,0],[0,500],[250,500]])

Ig = cv2.cvtColor(I,cv2.COLOR_BGR2GRAY)

H = homography_svd(srcpt,dstpt)


Itrnfd = cv2.warpPerspective(Ig,H,(Ig.shape[1],Ig.shape[0]))

plt.imshow(Itrnfd)

plt.title("Transformed Image")

plt.axis('off')

plt.waitforbuttonpress()
```

Output



Observations :

The obtained image is observed to have 2D perspective while original image has 3D perspective.

Conclusion:-

As the experiment performed,

- homography matrix was estimated using singular value decomposition as well as using direct linear transformation methods in python language.
- the homography matrix translates the 3D perspective of the image into 2D form which is tested on an image and 2D transformed perspective image is obtained.

Libraries and functions used are matplotlib, OpenCV, numpy, warpPerspective().