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

Roll No. : 21BEC027

Experiment…4 – Perspective Transformation

Objective : Remove the projective distortion in the image using homography.

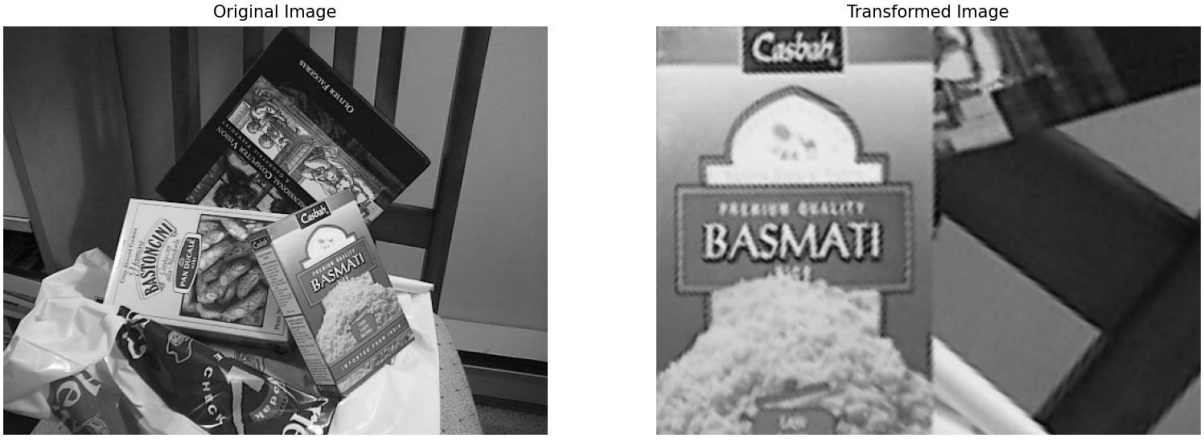
Import necessary libraries…

```
'''
Created on 2 September 2024 Mon 3:15:22 pm...
'''
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Task 1. For perspective transformation, you need a 3x3 transformation matrix. Straight lines will remain straight even after the transformation. To find this transformation matrix, you need 4 points on the input image and corresponding points on the output image. Among these 4 points, 3 of them should not be collinear. Then the transformation matrix can be found by the function *cv2.getPerspectiveTransform()*. Then apply *cv2.warpPerspective()* with this 3x3 transformation matrix.

```
#Task 1
I = cv2.imread(r"D:\Nirma Files\Computer Vision\box_sample.png")
rows,cols,ch = I.shape
pt1 = np.float32([[247,190],[325,154],[319,347],[386,293]])
pt2 = np.float32([[0,0],[256,0],[0,512],[256,512]])
matrix_aff = cv2.getPerspectiveTransform(pt1,pt2)
dst = cv2.warpPerspective(I,matrix_aff,(cols,rows))
plt.subplot(121),plt.imshow(I),plt.title("Original Image")
plt.axis("off")
plt.subplot(122),plt.imshow(dst),plt.title("Transformed Image")
plt.axis("off")
plt.show()
```

Output



Observation :

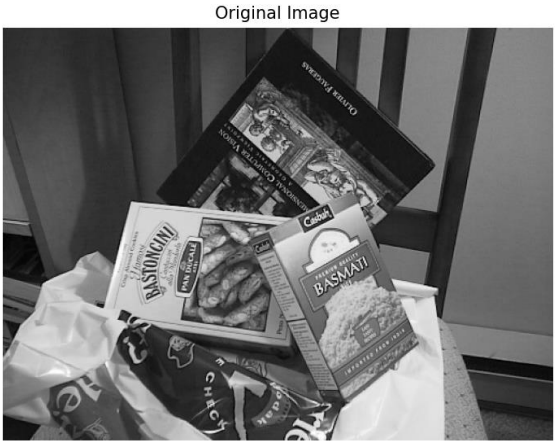
- The homography matrix is calculated using built-in function getPerspectiveTransform().
- Here, the output image is observed to be removed from projective distortion w.r.t. to original image which has projective distortion in 2D form.

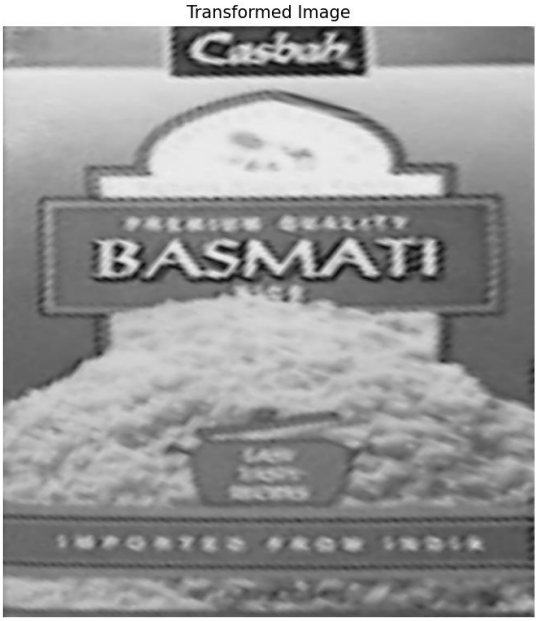
Task 2. Compute homography matrix for 4 given points using DLT and transform the point using computed homography matrix.

```
#Task 2...

print("Perspective Transform function")


def perspective_transform(img,x1,y1,x2,y2,x3,y3,x4,y4,width,height):
    pts_src = np.array([[x1,y1],[x2,y2],[x3,y3],[x4,y4]],dtype = np.float32)
    pts_dst = np.array([[0,0],[width-1,0],[0,height-1],[width-1,height-1]],dtype = np.float32)
    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)
    trnfd_img = cv2.warpPerspective(img,H,(width,height))
    return trnfd_img
```

```
Itrnfd = perspective_transform(I,247,190,325,154,319,347,386,293,900,1000)
plt.subplot(121),plt.imshow(I),plt.title("Original Image")
plt.axis("off")
plt.subplot(122),plt.imshow(Itrnfd),plt.title("Transformed Image")
plt.axis("off")
plt.show()
```

Output





Observation :

- The homography matrix is calculated using DLT algorithm with SVD such that the dimensions of output image can be set by user itself.
- Here, the rows and columns parameters are manually given in the homography matrix in order to calculate desired homography degree of freedoms.

Conclusion:-

As the experiment performed,

- homography matrix was estimated using built-in functions of OpenCV library.
- the homography matrix was implemented on images to remove projective distortions in 2D images.

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