11. Perform Affine Transformation on the image.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/91824/Downloads/bird.jpeg")
rows,cols,_ = img.shape
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv2.getAffineTransform(pts1,pts2)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow("Affine Transform", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :

12. Perform Perspective Transformation on the image.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/91824/Downloads/eswar.jpeg")
rows,cols,ch = img.shape
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[100,50],[300,0],[0,300],[300,300]])
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(img,M,(cols, rows))
cv2.imshow('Transformed Image', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

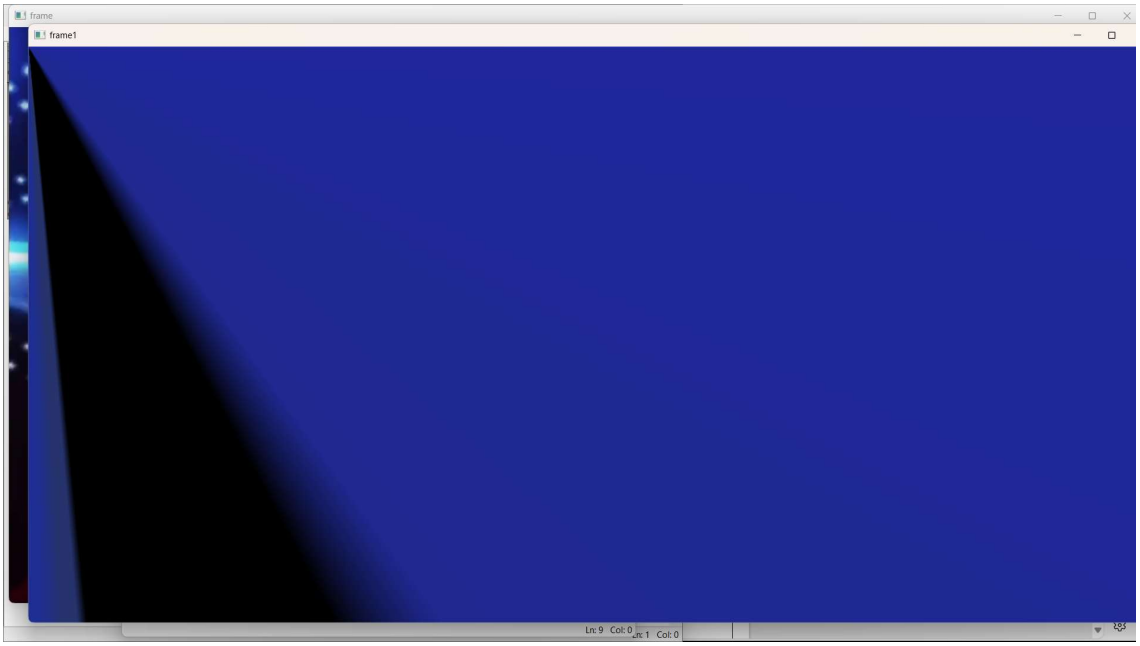
OUTPUT :

13. Perform Perspective Transformation on the Video.

PROGRAM :

```
import cv2

import numpy as np

cap = cv2.VideoCapture("C:/Users/91824/Videos/Hanuman 2024 Telugu HDTS 1080p x264 AAC HC-ESub CineVood.mkv")

while True:

 ret, frame = cap.read()

    pts1 = np.float32([[200,300], [5, 2],[0, 4], [6, 0]])

    pts2 = np.float32([[0, 0], [4, 0],[0, 1], [4, 6]])

    matrix = cv2.getPerspectiveTransform(pts1, pts2)

    result = cv2.warpPerspective(frame, matrix, (0, 0))

    cv2.imshow('frame', frame) # Initial Capture

    cv2.imshow('frame1', result) # Transformed Capture

    if cv2.waitKey(24) == 27:

        break

cap.release()

cv2.destroyAllWindows()
```

INPUT:

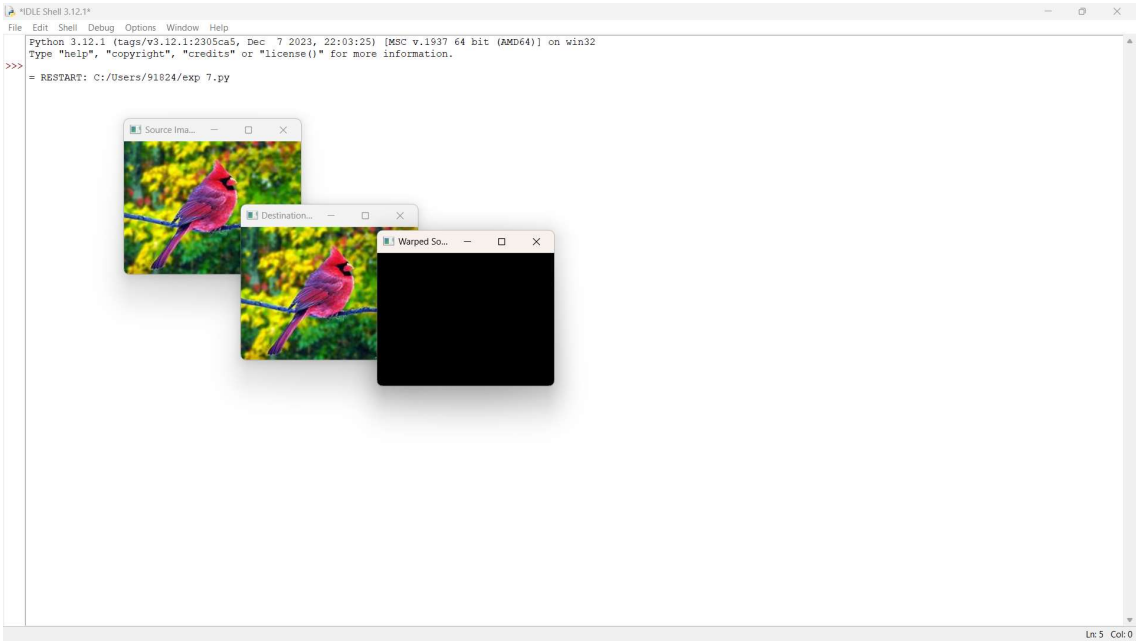


14. Perform transformation using Homography matrix

PROGRAM:

```
import cv2
import numpy as np
im_src = cv2.imread("C:/Users/91824/Downloads/trees.jpeg")
pts_src = np.array([[141, 131], [480, 159], [493, 630],[64, 601]])
im_dst = cv2.imread("C:/Users/91824/Downloads/trees.jpeg")
pts_dst = np.array([[318, 256],[534, 372],[316, 670],[73, 473]])
h, status = cv2.findHomography(pts_src, pts_dst)
im_out = cv2.warpPerspective(im_src, h, (im_dst.shape[1],im_dst.shape[0]))
cv2.imshow("Source Image", im_src)
cv2.imshow("Destination Image", im_dst)
cv2.imshow("Warped Source Image", im_out)
cv2.waitKey(0)
```

OUTPUT:

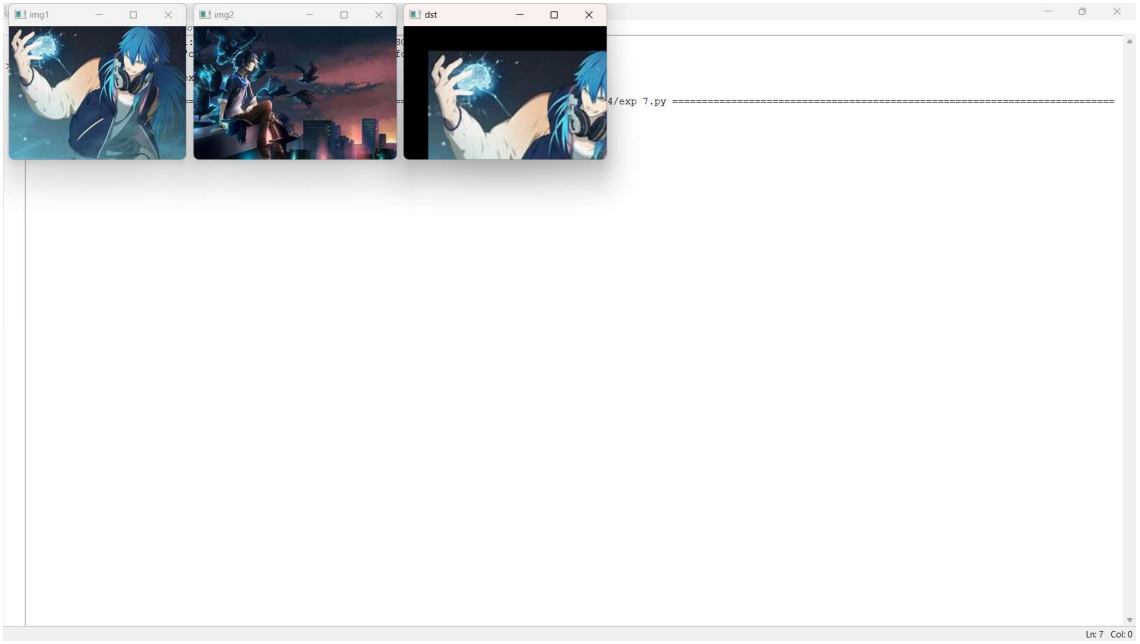


15. Perform transformation using Direct Linear Transformation

PROGRAM :

```
import cv2
import numpy as np
img1 = cv2.imread("C:/Users/91824/OneDrive/affilitate.jpeg")
img2 = cv2.imread("C:/Users/91824/OneDrive/anime.jpeg")
pts1 = np.array([[50, 50], [200, 50], [50, 200], [200, 200]])
pts2 = np.array([[100, 100], [300, 100], [100, 300], [300, 300]])
H, _ = cv2.findHomography(pts1, pts2)
dst = cv2.warpPerspective(img1, H, (img2.shape[1], img2.shape[0]))
cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('dst', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:



16. Perform Edge detection using canny method

PROGRAM:

```
import cv2
img = cv2.imread(r"C:\Users\91824\OneDrive\sword.jpeg")
cv2.imshow('Original', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge Detection
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT :

OUTPUT :

17. Perform Edge detection using Sobel Matrix along X axis

PROGRAM:

```
import cv2
img = cv2.imread(r"C:\Users\91824\OneDrive\LUFFY.JPEG")
cv2.imshow('Original', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)
cv2.imshow('Sobel X', sobelx)
cv2.waitKey(0)
```

INPUT :

OUTPUT:

18. Perform Edge detection using Sobel Matrix along Y axis

PROGRAM:

```
import cv2
img = cv2.imread(r"C:\Users\91824\OneDrive\onepiece.jpg")
cv2.imshow('Original', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)
cv2.imshow('Sobel Y', sobely)
cv2.waitKey(0)
```

INPUT:

OUTPUT:

19. Perform Edge detection using Sobel Matrix along XY axis

PROGRAM:

```
import cv2
img = cv2.imread(r"C:\Users\91824\OneDrive\trees.jpeg")
cv2.imshow('Original', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)
cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
cv2.waitKey(0)
```

INPUT :

OUTPUT:

20. Perform Sharpening of Image using Laplacian mask with negative center coefficient.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread(r"C:\Users\91824\OneDrive\nature.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = np.array([[0,1,0], [1,-8,1], [0,1,0]])
sharpened = cv2.filter2D(gray, -1, kernel)
cv2.imshow('Original', gray)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT:

OUTPUT: