

13. Perform Perspective Transformation on the Video

PROGRAM:

```
import cv2

import numpy as np

cap = cv2.VideoCapture(r"C:\Users\vempa\Videos\test.mp4")

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

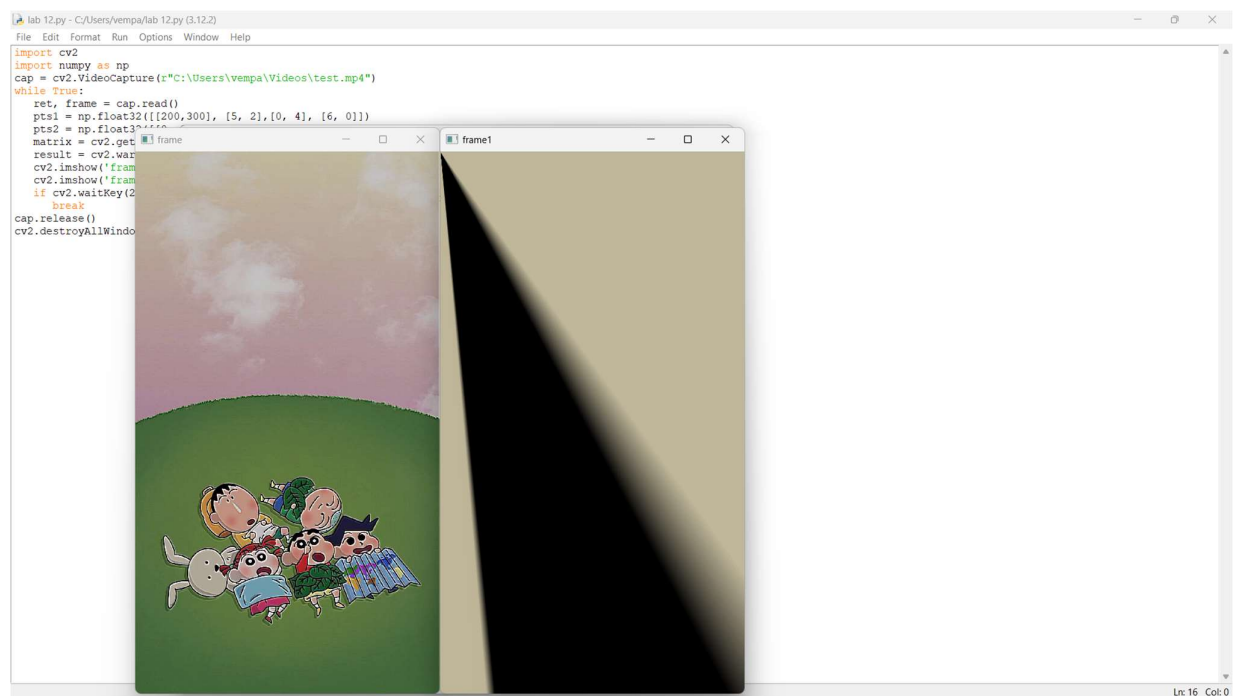
    if cv2.waitKey(24) == 27:

        break

cap.release()

cv2.destroyAllWindows()
```

OUTPUT:



14. Perform transformation using Homography matrix

PROGRAM:

```
import cv2

import numpy as np

im_src = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")
pts_src = np.array([[141, 131], [480, 159], [493, 630], [64, 601]])
im_dst = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")
pts_dst = np.array([[318, 256], [534, 372], [316, 670], [73, 473]])

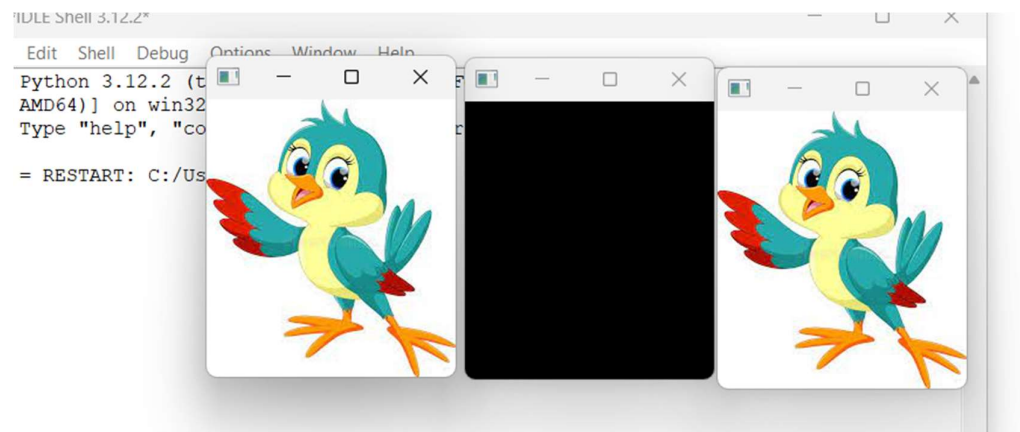
h, status = cv2.findHomography(pts_src, pts_dst)

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

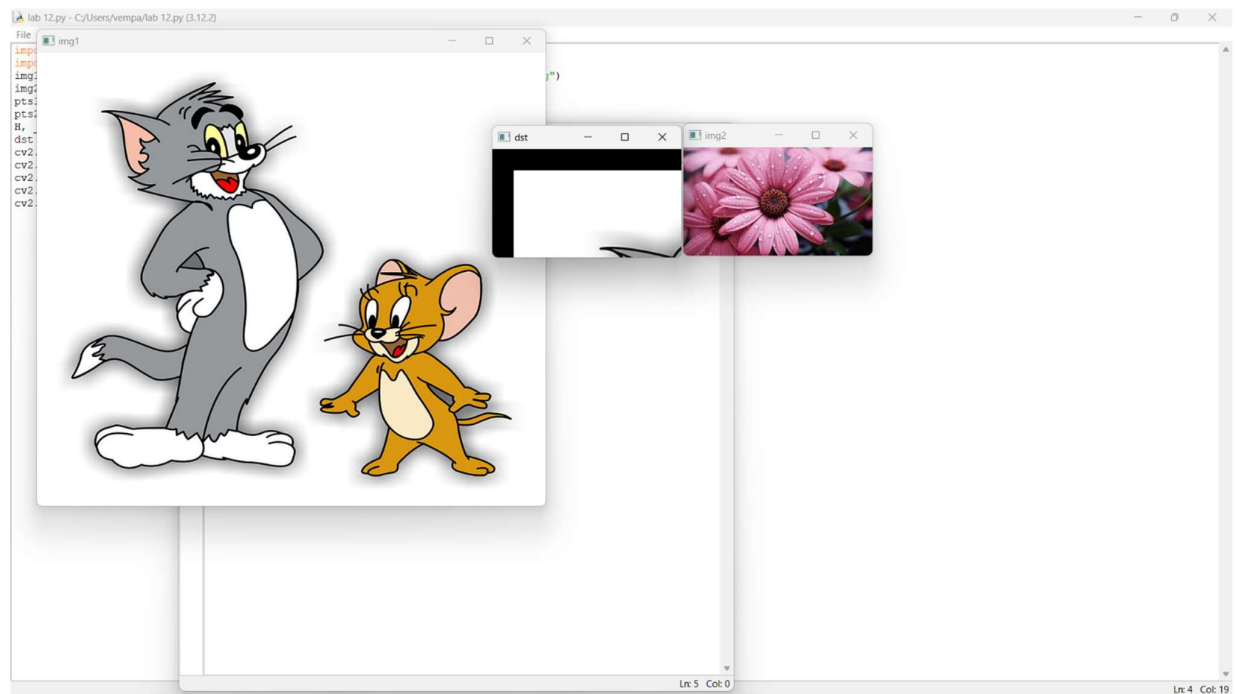
```
import cv2

import numpy as np

img1 = cv2.imread(r"C:/Users/vempa/Downloads/HD-wallpaper-tom-and-jerry-cartoons.jpg")
img2 = cv2.imread(r"C:\Users\vempa\Downloads\LAB4.jpg")
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

# Read the input image
image_path = r"C:\Users\vempa\Downloads\BIRD2.jpg"
original_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Check if the image is successfully loaded
if original_image is None:
    print("Error: Could not load the image.")
else:
    # Apply Gaussian blur to reduce noise and improve edge detection
    blurred_image = cv2.GaussianBlur(original_image, (5, 5), 0)

    # Apply Canny edge detection
    edges = cv2.Canny(blurred_image, 50, 150) # Adjust the threshold values as needed

    # Display the original image and the result
    cv2.imshow("Original Image", original_image)
    cv2.imshow("Canny Edge Detection", edges)

    cv2.waitKey(0)

    cv2.destroyAllWindows()
```

OUTPUT:

```
import cv2

# Read the input image
image_path = r"C:\Users\vempa\Downloads\BIRD2.jpg"
original_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


# Check if the image is successfully loaded
if original_image is None:
    print("Error: Could not load the image.")
else:
    # Apply Gaussian blur to reduce noise and improve edge detection
    blurred_image = cv2.GaussianBlur(original_image, (5, 5), 0)

    # Apply Canny edge detection
    edges = cv2.Canny(blurred_image, 50, 150) # Adjust the threshold values as needed

    # Display the original image and the result
    cv2.imshow("Original Image", original_image)
    cv2.imshow("Canny Edge Detection", edges)

    cv2.waitKey(0)

    cv2.destroyAllWindows()
```



17. Perform Edge detection using Sobel Matrix along X axis

PROGRAM:

```
import cv2

img = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")

cv2.imshow('Original', img)

cv2.waitKey(0)

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur the image for better edge detection

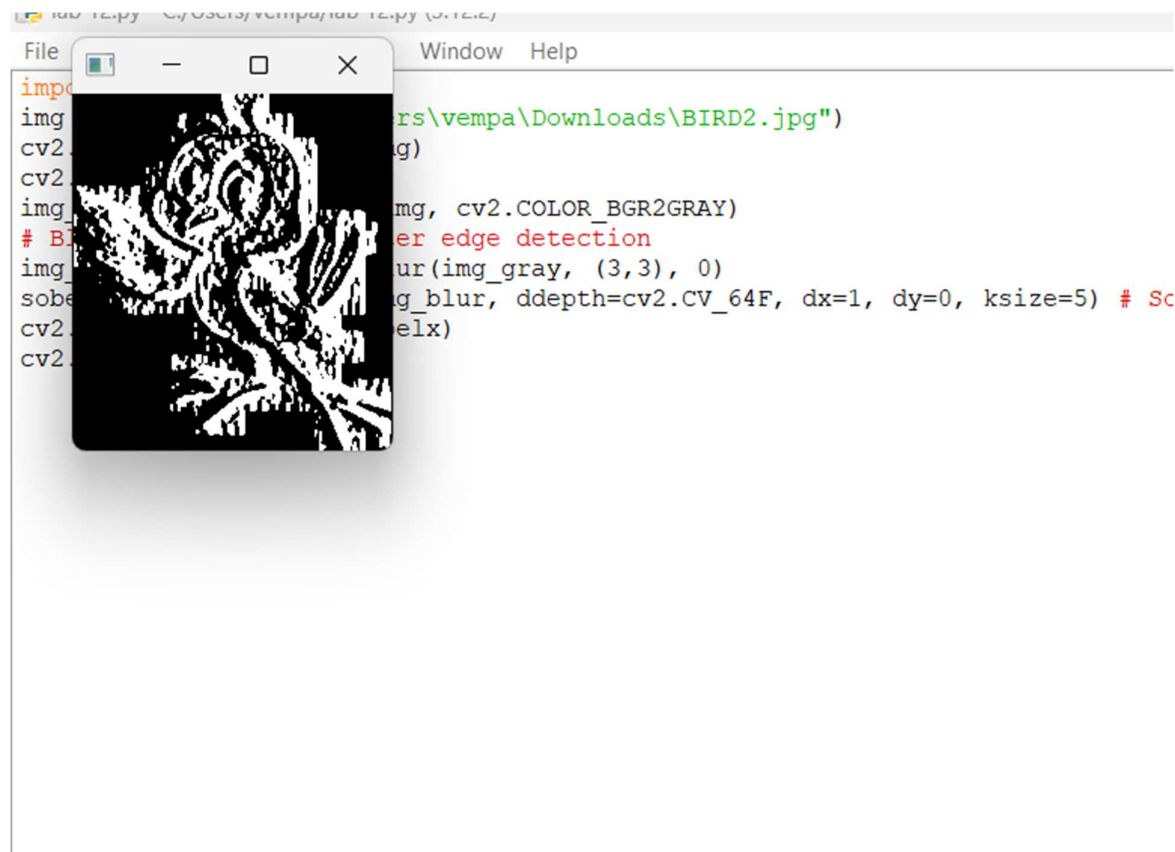
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X
axis

cv2.imshow('Sobel X', sobelx)

cv2.waitKey(0)
```

OUTPUT:



18. Perform Edge detection using Sobel Matrix along Y axis

PROGRAM:

```
import cv2

# Read the original image

img = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")

# Display original image

cv2.imshow('Original', img)

cv2.waitKey(0)

# Convert to grayscale

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur the image for better edge detection

img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

# Sobel Edge Detection

sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge Detection on the Y
axis

# Display Sobel Edge Detection Images

cv2.imshow('Sobel Y', sobely)

cv2.waitKey(0)
```

OUTPUT:



19. Perform Edge detection using Sobel Matrix along XY axis

PROGRAM :

```
import cv2

img = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")

# Display original image
cv2.imshow('Original', img)

cv2.waitKey(0)

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur the image for better edge detection
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) # Combined X and Y Sobel Edge
Detection

cv2.imshow('Sobel X Y using Sobel() function', sobelxy)

cv2.waitKey(0)
```

OUTPUT:

```
ng_gray = cv
Blur the im
ng_blur = cv
obelxy = cv2
v2.imshow('S
v2.waitKey(0 >>>

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb
AMD64)] on win32
Type "help", "copyright", "credits" or "l:
===== RESTART: C:/Users,
```



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\vempa\Downloads\HD-wallpaper-tom-and-jerry-cartoons.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()
```

OUTPUT:



