opencv

August 26, 2024

[11]:

```
!pip install opencv-python

import cv2

# Read the image from fig
img = cv2.imread('C:
 ↪\\Users\\moham\\OneDrive\\Desktop\\Documents\\Pictures\\sana1.jpg')

# Display the image in a window
cv2.imshow('Image', img)

# Wait for a key press and close the image window
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Requirement already satisfied: opencv-python in
c:\users\moham\anaconda3\lib\site-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in
c:\users\moham\anaconda3\lib\site-packages (from opencv-python) (1.24.3)
```

[2]:

```
# Resize the image to a specific width and height
resized_img = cv2.resize(img, (400, 400))
# Display the resized image
cv2.imshow('Resized Image', resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[3]:

```
# Convert the image to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Display the grayscale image
cv2.imshow('Grayscale Image', gray_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[4]:

```
# Apply Gaussian blur to the image
blurred_img = cv2.GaussianBlur(img, (15, 15), 0)
```

```
# Display the blurred image
cv2.imshow('Blurred Image', blurred_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[5]:

```
# Perform Canny edge detection
edges = cv2.Canny(img, 100, 200)

# Display the edges
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[6]:

```
# Apply binary thresholding
_, thresholded_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY)

# Display the thresholded image
cv2.imshow('Thresholded Image', thresholded_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[7]:

```
# Draw a rectangle on the image
rect_img = img.copy()
cv2.rectangle(rect_img, (50, 50), (200, 200), (0, 255, 0), 3)

# Draw a circle on the image
cv2.circle(rect_img, (300, 300), 50, (255, 0, 0), -1)

# Display the image with shapes
cv2.imshow('Image with Shapes', rect_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[8]:

```
# Load the Haar Cascade for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +␣
 ↪'haarcascade_frontalface_default.xml')

# Convert the image to grayscale (Haar Cascade works on grayscale images)
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = face_cascade.detectMultiScale(gray_img, scaleFactor=1.1,␣
 ↪minNeighbors=5, minSize=(30, 30))

# Draw rectangles around the detected faces
for (x, y, w, h) in faces:
```

```python
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

# Display the image with faces detected
cv2.imshow('Faces Detected', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[9]:

```python
# Split the image into its blue, green, and red channels
b, g, r = cv2.split(img)

# Display the individual channels
cv2.imshow('Blue Channel', b)
cv2.imshow('Green Channel', g)
cv2.imshow('Red Channel', r)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[10]:

```python
# Get the image dimensions
(h, w) = img.shape[:2]

# Calculate the center of the image
center = (w // 2, h // 2)

# Define the rotation matrix
M = cv2.getRotationMatrix2D(center, 45, 1.0)

# Rotate the image
rotated_img = cv2.warpAffine(img, M, (w, h))

# Display the rotated image
cv2.imshow('Rotated Image', rotated_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[ ]:

[ ]:

[ ]: