1

import cv2

img = cv2.imread('image.jpg')

print("Image:", img.dtype, img.shape, img.size)

cv2.imshow('Image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

cap = cv2.VideoCapture('video.mp4')

while cap.isOpened():

    ret, frame = cap.read()

    if not ret: break

    print("Video Frame:", frame.dtype, frame.shape, frame.size)

    cv2.imshow('Video', frame)

    if cv2.waitKey(25) & 0xFF == ord('q'): break

cap.release()

cv2.destroyAllWindows()

2

import cv2

img = cv2.imread('input.jpg')

cv2.imwrite('output.jpg', img)

print("Image read and saved successfully.")

cap = cv2.VideoCapture(0)

print("Press 'q' to quit the webcam window.")

while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        break

  cv2.imshow('Webcam Video', frame)

If cv2.waitKey(1) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()

3

import cv2

img = cv2.imread('image.jpg')

print("Before:")

print(f"{'Property':<10} {'Value'}")

print(f"{'Shape':<10} {img.shape}")

print(f"{'Size':<10} {img.size}")

print(f"{'Dtype':<10} {img.dtype}")

cv2.imshow("Original Image", img)

resized = cv2.resize(img, (200, 200))

cv2.imshow("Resized Image", resized)

cropped = img[50:200, 100:300]

cv2.imshow("Cropped Image", cropped)

print("\nAfter (Resized):")

print(f"{'Shape':<10} {resized.shape}")

print(f"{'Size':<10} {resized.size}")

print(f"{'Dtype':<10} {resized.dtype}")

print("\nAfter (Cropped):")

print(f"{'Shape':<10} {cropped.shape}")

print(f"{'Size':<10} {cropped.size}")

print(f"{'Dtype':<10} {cropped.dtype}")

cv2.waitKey(0)

cv2.destroyAllWindows()

4

import cv2

img = cv2.imread('image.jpg')

cv2.imshow("Original (BGR)", img)

print("Original Image:")

print("Color Space: BGR")

print("Shape:", img.shape)

print("Size:", img.size)

print("Dtype:", img.dtype)

rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

cv2.imshow("RGB Image", rgb)

print("\nRGB Image:")

print("Shape:", rgb.shape)

print("Size:", rgb.size)

print("Dtype:", rgb.dtype)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow("Grayscale Image", gray)

print("\nGrayscale Image:")

print("Shape:", gray.shape)

print("Size:", gray.size)

print("Dtype:", gray.dtype)

hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

cv2.imshow("HSV Image", hsv)

print("\nHSV (MVT) Image:")

print("Shape:", hsv.shape)

print("Size:", hsv.size)

print("Dtype:", hsv.dtype)

cv2.waitKey(0)

cv2.destroyAllWindows()

5

import cv2

import numpy as np

def add\_salt\_pepper\_noise(img, salt=0.02, pepper=0.02):

    noisy = img.copy()

    h, w = img.shape[:2]

    # Salt noise (white)

    for \_ in range(int(salt \* h \* w)):

        x, y = np.random.randint(0, w), np.random.randint(0, h)

        noisy[y, x] = 255

    # Pepper noise (black)

    for \_ in range(int(pepper \* h \* w)):

        x, y = np.random.randint(0, w), np.random.randint(0, h)

        noisy[y, x] = 0

    return noisy

img = cv2.imread('image.jpg')

sp\_img = add\_salt\_pepper\_noise(img)

# Blurring on normal image

blur1 = cv2.blur(img, (5,5))

blur2 = cv2.medianBlur(img, 5)

blur3 = cv2.GaussianBlur(img, (5,5), 0)

blur4 = cv2.bilateralFilter(img, 9, 75, 75)

# Blurring on noisy image

sp\_blur1 = cv2.blur(sp\_img, (5,5))

sp\_blur2 = cv2.medianBlur(sp\_img, 5)

sp\_blur3 = cv2.GaussianBlur(sp\_img, (5,5), 0)

sp\_blur4 = cv2.bilateralFilter(sp\_img, 9, 75, 75)

cv2.imshow('Original', img)

cv2.imshow('Salt-Pepper Noise', sp\_img)

cv2.imshow('Blur Average', blur1)

cv2.imshow('Median Blur', blur2)

cv2.imshow('Gaussian Blur', blur3)

cv2.imshow('Bilateral Filter', blur4)

cv2.imshow('Noisy Blur Average', sp\_blur1)

cv2.imshow('Noisy Median Blur', sp\_blur2)

cv2.imshow('Noisy Gaussian Blur', sp\_blur3)

cv2.imshow('Noisy Bilateral Filter', sp\_blur4)

6

import cv2

img1 = cv2.imread('coins.jpg')

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

\_, th\_bin = cv2.threshold(gray1, 127, 255, cv2.THRESH\_BINARY)

\_, th\_bin\_inv = cv2.threshold(gray1, 127, 255, cv2.THRESH\_BINARY\_INV)

cv2.imshow('Simple Threshold Binary', th\_bin)

cv2.imshow('Simple Threshold Binary Inv', th\_bin\_inv)

img2 = cv2.imread('handwritten.jpg')

gray2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

adap\_bin = cv2.adaptiveThreshold(gray2, 255,

                                 cv2.ADAPTIVE\_THRESH\_MEAN\_C,

                                 cv2.THRESH\_BINARY, 11, 2)

adap\_bin\_inv = cv2.adaptiveThreshold(gray2, 255,

                                     cv2.ADAPTIVE\_THRESH\_MEAN\_C,

                                     cv2.THRESH\_BINARY\_INV, 11, 2)

cv2.imshow('Adaptive Threshold Binary', adap\_bin)

cv2.imshow('Adaptive Threshold Binary Inv', adap\_bin\_inv)

7

import cv2

import numpy as np

img = np.zeros((400, 400, 3), dtype=np.uint8)

img.fill(255)

cv2.circle(img, (100, 100), 50, (255, 0, 0), -1)

cv2.line(img, (50, 300), (350, 300), (0, 255, 0), 5)

cv2.rectangle(img, (200, 50), (350, 150), (0, 0, 255), -1)  # filled rectangle

cv2.putText(img, 'YourName123', (10, 390), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 0), 2)

cv2.imshow('Drawing', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite('drawn\_image.jpg', img)

8

import cv2

img = cv2.imread('object.jpg', cv2.IMREAD\_GRAYSCALE)

edges = cv2.Canny(img, threshold1=100, threshold2=200)

cv2.imshow('Original', img)

cv2.imshow('Edges - Canny', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

9

import cv2

img = cv2.imread('objects.jpg', cv2.IMREAD\_GRAYSCALE)

\_, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

img\_color = cv2.cvtColor(img, cv2.COLOR\_GRAY2BGR)

cv2.drawContours(img\_color, contours, -1, (0,255,0), 2)

cv2.imshow('Contours', img\_color)

cv2.waitKey(0)

cv2.destroyAllWindows()

12

import cv2

import numpy as np

img = cv2.imread('image.jpg', 0)

edges = cv2.Canny(img, 50, 150)

lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, minLineLength=100, maxLineGap=10)

if lines is not None:

  for line in lines:

    x1, y1, x2, y2 = line[0]

    cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 2)

corners = cv2.cornerHarris(img, 2, 3, 0.04)

img[corners > 0.01 \* corners.max()] = [0, 0, 255]

cv2.imshow('Result', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

13

import cv2

def match\_keypoints(img1\_path, img2\_path):

  img1 = cv2.imread(img1\_path, 0)

  img2 = cv2.imread(img2\_path, 0)

  sift = cv2.SIFT\_create()

  kp1, des1 = sift.detectAndCompute(img1, None)

  kp2, des2 = sift.detectAndCompute(img2, None)

  bf = cv2.BFMatcher()

  matches = bf.knnMatch(des1, des2, k=2)

  good\_matches = [m for m, n in matches if m.distance < 0.75 \* n.distance]

  img\_matches = cv2.drawMatches(img1, kp1, img2, kp2, good\_matches, None, flags=2)

  cv2.imshow("Matches", img\_matches)

  cv2.waitKey(0)

  cv2.destroyAllWindows()

match\_keypoints('image1.jpg', 'image2.jpg')

14

import cv2

import numpy as np

def match\_transposed\_features(image\_path):

  img = cv2.imread(image\_path, 0)

  img\_t = np.transpose(img)

  sift = cv2.SIFT\_create()

  kp1, des1 = sift.detectAndCompute(img, None)

  kp2, des2 = sift.detectAndCompute(img\_t, None)

  bf = cv2.BFMatcher()

  matches = bf.knnMatch(des1, des2, k=2)

  good = [m for m, n in matches if m.distance < 0.75 \* n.distance]

  match\_img = cv2.drawMatches(img, kp1, img\_t, kp2, good, None, flags=2)

  cv2.imshow("Matches", match\_img)

  cv2.waitKey(0)

  cv2.destroyAllWindows()

match\_transposed\_features('image.jpg')

10

import cv2

import numpy as np

image = cv2.imread('your\_image.jpg')

if image is None:

    print("Error: Image not found. Make sure 'your\_image.jpg' exists.")

    exit()

hsv\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

lower\_blue = np.array([100, 150, 50])

upper\_blue = np.array([140, 255, 255])

mask = cv2.inRange(hsv\_image, lower\_blue, upper\_blue)

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in contours:

   x, y, w, h = cv2.boundingRect(contour)

cv2.drawContours(image, [contour], -1, (0, 255, 0), 2)

cv2.putText(image, 'Blue', (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

cv2.imshow('Detected Blue Objects', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

11

import cv2

import numpy as np

image\_path = 'your\_image.jpg'

image = cv2.imread(image\_path)

if image is None:

    print(f"Error: Could not read the image. Make sure the file '{image\_path}' is in the correct path.")

    exit()

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

gray\_image = np.float32(gray\_image)

corner\_scores = cv2.cornerHarris(gray\_image, blockSize=2, ksize=3, k=0.04)

threshold = 0.01 \* corner\_scores.max()

image[corner\_scores > threshold] = [0, 0, 255]

cv2.imshow('Detected Corners', image)

cv2.waitKey(0)

cv2.destroyAllWindows()