Training Report Day-26

5 July 2024

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of tools and functions for processing images and videos, making it a powerful tool for computer vision applications like face detection, object tracking, edge detection, and more. OpenCV is widely used in applications ranging from real-time image processing to advanced machine learning models in computer vision.

Here's a guide to using OpenCV in Python, including how to install it, read and display images, basic operations, and some common applications.

1. Installing OpenCV

You can install OpenCV via pip:

pip install opencv-python

pip install opencv-python-headless # For environments without GUI support

2. Reading and Displaying Images

OpenCV provides functions to read, display, and save images.

```
import cv2

# Read an image

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

# Display the image in a window

cv2.imshow('Image', image)



# Wait until a key is pressed, then close the window
```

```
cv2.waitKey(0)

cv2.destroyAllWindows()
```

3. Basic Image Operations

3.1 Converting to Grayscale

Convert the image to grayscale, which is often useful for tasks like edge detection or thresholding.

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Grayscale Image', gray_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

4. Image Processing Techniques

4.1 Blurring

Blurring is useful for reducing noise in an image.

```
# Applying Gaussian Blur

blurred_image = cv2.GaussianBlur(image, (15, 15), 0)

cv2.imshow('Blurred Image', blurred_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

edges = cv2.Canny(gray_image, 100, 200)

cv2.imshow('Edges', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

5. Video Processing

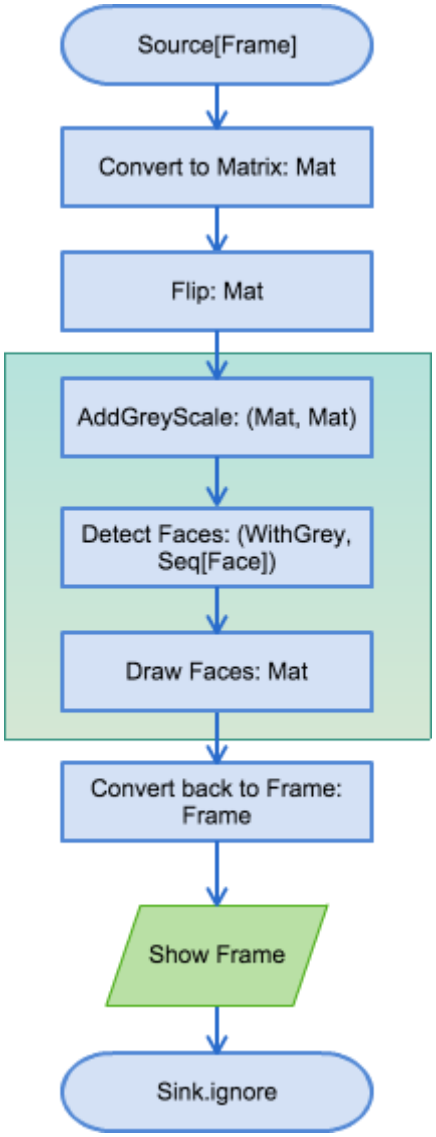
OpenCV can also capture and process video streams, either from a file or from a camera.

```
# Open video capture (0 for default camera)
cap = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    if not ret:
        break

    # Convert to grayscale
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame
    cv2.imshow('Video Stream', gray_frame)
    # Press 'q' to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release the capture and close windows
cap.release()
cv2.destroyAllWindows()
```


Source[Frame]
Convert to Matrix: Mat
Flip: Mat
AddGreyScale: (Mat, Mat)
Detect Faces: (WithGrey, Seq[Face])
Draw Faces: Mat
Convert back to Frame: Frame
Show Frame
Sink.ignore