OpenCV Commands

Name - Prathiksha Rao
ADR

1. cv2.imread() - It is used to read an image from the specified file path. First Argument - file path to the image you want to read.
Prefix the string with 'r' to create a raw string literal. This will tell Python to interpret the string as-is without processing escape characters.
Second Argument - specifies how the image should be read.
   a. cv2.IMREAD_GRAYSCALE - image should be read as grayscale.
   b. cv2.IMREAD_COLOR - read the image in color.
   c. cv2.IMREAD_UNCHANGED - read the image as it is (including alpha channel, if present).

2. cv2.imshow() - This function is used to display an image in a window. First Argument - title of the window. Second Argument - image you want to display.

3. cv2.waitKey() - This function is used to wait for a keyboard event. It pauses the execution of the program until a key is pressed. The argument is the time to wait for a key event in milliseconds. Passing '0' means that the program will wait indefinitely until a key is pressed.

4. cv2.destroyAllWindows() - This function is used to close all OpenCV windows.
It's typically called after cv2.waitKey() to ensure that all windows are closed properly when the program exits.

5. plt.imshow() - This function is from the matplotlib.pyplot module. It is used to display an image.The cmap parameter specifies the colormap to be used for displaying the image. In this case, 'gray' indicates grayscale colormap.
Interpolation is used to estimate pixel values at positions between existing pixels. Bicubic interpolation is a method used in image processing to interpolate pixel values when scaling or transforming an image. It's an extension of the cubic interpolation method and provides smoother results compared to linear interpolation.

6. plt.plot() - This function is used to plot lines or markers on a graph.
7. plt.show() - This function is used to display the plot or image.

8. cv2.VideoCapture() - function from the OpenCV library that is used to capture video from cameras, video files, or image sequences. Used to read video frames. For using the default camera 0 is used , 1 for external cameras and so on.

9. cv2.VideoWriter_fourcc(*'XVID') -
   a. What is a codec? A codec is a piece of software that encodes or decodes digital video and audio data. The term codec is a combination of "compressor-decompressor" or "coder-decoder." Codecs are essential for reducing the size of large video files while preserving quality and ensuring efficient playback.

b. FourCC, which stands for "Four Character Code," is a sequence of four bytes (typically ASCII characters) used to uniquely identify data formats, such as video codecs, video formats, and pixel formats. Video processing with OpenCV and other multimedia frameworks, FourCC codes are used to specify the codec to be used for encoding and decoding video streams.
   The code is made up of four characters, each representing one byte. For example, 'XVID', 'MJPG', and 'H264' are common FourCC codes.
c. cv2.VideoWriter_fourcc() to define the codec for writing video files. This is essential for ensuring that the video is encoded in a format that can be played back correctly.
d. The *'XVID' syntax is used to pass the string as separate characters to the function.

10. cv2.VideoWriter() - is a class in the OpenCV library that is used to write video files from a sequence of image frames. It essentially creates a video file container and allows you to add frames to it, resulting in a playable video file.

11. Ret, frame=cap.read() - It reads a single frame from the video capture source (such as a webcam) and returns two values - Ret: A boolean value indicating whether the frame was successfully read. If the frame was read successfully, ret is True; otherwise, it's False. Frame: The actual frame data that was captured from the video source. This frame is typically represented as a NumPy array.

12. cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY): This converts the captured frame from BGR color space (used by OpenCV) to grayscale.
cap.release(): This releases the video capture object, freeing up any resources it was using.
out.release(): This releases the video writer object, ensuring that the video file is properly saved and closed.

15. cv2.line( ) - draw a line on an image, cv2.rectangle() , cv2.circle(), cv2.polylines(), cv2.putText().

16. cv2.add(img1,img2) #adds the pixel values of the two images
cv2.addWeighted(img1,0.6,img2,0.4,0) #superimpose images on each other
17. cv2.threshold() - Thresholding is a fundamental technique in image processing that helps in converting a grayscale image into a binary image by applying a threshold value. This is useful for separating objects in an image from the background or for simplifying the image for further analysis.
A. cv2.THRESH_BINARY - This method involves setting a single threshold value. All pixel values above this threshold are set to the maximum value (often 255 for white), and all values below are set to the minimum value (often 0 for black).
B. cv2.THRESH_BINARY_INV - This is similar to simple thresholding but inverts the output.
C. cv2.THRESH_TRUNC - Pixels above the threshold value are set to the threshold value, while those below remain unchanged.
D. cv2.THRESH_TRUNC - Pixels above the threshold value are set to the threshold value, while those below remain unchanged.

18. Bitwise operations are used to manipulate binary data at the level of individual bits. These operations can be applied to images for tasks like masking, combining images, and creating effects. cv2.bitwise_and() , cv2.bitwise_or , cv2.bitwise_not.

19. cv2.ADAPTIVE_THRESH_GAUSSIAN_C: Adaptive thresholding method that calculates the threshold for a pixel based on a weighted sum (Gaussian mean) of its neighbors.

cv2.THRESH_BINARY + cv2.THRESH_OTSU: Combines binary thresholding with Otsu's method.
otsu: Output binary image after applying Otsu's binarization.
retval2: Optimal threshold value found by Otsu's method.

20. cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
HSV stands for Hue, Saturation, and Value.
Hue: Represents the color type. Different colors are represented by different angles.
(e.g., red is around 0 or 180, green is around 60, and blue is around 120).
Saturation: Measures the vibrancy of the color. It ranges from 0 (gray) to 255 (fully saturated color).
Value: Measures the brightness of the color. It ranges from 0 (black) to 255 (maximum brightness).

The HSV color space is often preferred over BGR for tasks involving color manipulation and segmentation because: Hue is less sensitive to lighting changes and shadows, making it easier to separate colors based on their type rather than their intensity. Saturation and Value provide control over the vibrancy and brightness, which can be useful for image enhancement and filtering.

21. cv2.inRange() Function: checks if the pixel values of an image fall within a specified range.

22. cv2.GaussianBlur() - Gaussian blur applies a Gaussian function (a bell-shaped curve) to the pixels in the image.

cv2.medianBlur()-Median blur replaces each pixel's value with the median value of the neighboring pixels defined by the kernel size.

cv2.bilateralFilter()- Bilateral filtering is a type of blur that preserves edges while smoothing other regions. It averages pixel values based on both spatial closeness and color similarity.

cv2.erode()-Erosion removes pixels on object boundaries. It effectively "shrinks" the white regions (foreground) in the image. It is used to remove small white noise, detach two connected objects, and reduce the size of objects in binary images.

cv2.dilate()-Dilation adds pixels to the boundaries of objects in an image, effectively "expanding" the white regions (foreground). It is used to enlarge objects, fill small holes, and connect broken parts in binary images.

cv2.morphologyEx()-cv2.MORPH_OPEN-Opening is a combination of erosion followed by dilation. It removes small objects or noise from the foreground. It is particularly effective at removing noise without affecting the shape and size of the objects significantly.

cv2.MORPH_CLOSE-Closing is a combination of dilation followed by erosion. It fills small holes or gaps within the foreground objects.

It is used to close small holes or connect small breaks in the objects while preserving their overall shape and size.

cv2.laplacian()- It highlights regions with rapid intensity changes. It is sensitive to noise and typically requires additional preprocessing steps or filtering.

cv2.Sobel()-Sobel operators compute the gradient of the image intensity. They emphasize edges by highlighting areas with high gradient magnitude.

cv2.Canny()-The Canny edge detector is a multi-step algorithm that first calculates gradients, then applies non-maximum suppression to thin edges, and finally performs edge tracking by hysteresis.

23. cv2.createBackgroundSubtractorMOG2() - This algorithm is commonly used for background subtraction in videos, particularly in scenarios where the background may vary over time or contain shadows.

24. ORB (Oriented FAST and Rotated BRIEF) is a feature detector and descriptor. orb.detectAndCompute()- detects keypoints and computes descriptors.

The BFMatcher class performs pairwise matching between descriptors from two sets of keypoints.

25. Detects corners in the grayscale image using the cv2.goodFeaturesToTrack() function.

26. cv2.grabCut() function in OpenCV is used for performing the GrabCut algorithm, which is a powerful technique for foreground/background segmentation in images.

27. cv2.matchTemplate() function in OpenCV is used to search for a template image within a larger image by computing the similarity between the template and different locations in the larger image. It works by sliding the template image over the larger image and computing a matching score at each position.