Vidyavardhini’s College of Engineering &
Technology Department of Computer Engineering

Aim: To perform Handling Files, Cameras and GUIs

Objective: To perform Basic I/O Scripts, Reading/Writing an Image File, Converting Between an Image and raw bytes, Accessing image data with numpy.array, Reading /writing a video file, Capturing camera, Displaying images in a window ,Displaying camera frames in a window

Theory:

Basic I/O script

A large number of CV applications require image input. Most of them also output images. A camera can be needed as an input source and a window as an output destination for an interactive CV application. Raw bytes, video files, and image files are other potential sources and destinations. For instance, if we use generative graphics in our application, raw bytes might be created by an algorithm and communicated over a network connection.

Reading/Writing an Image File

Reading An Image File:

For reading an image, the imread() function is used in OpenCV. Here’s the syntax:

imread(filename, flags)

It takes two arguments:

The first argument is the image name, which requires a fully qualified path name to the file.

The second argument is an optional flag that lets you specify how the image should be represented. OpenCV offers several options for this flag, but those that are most common include:

cv2.IMREAD_UNCHANGED or -1

cv2.IMREAD_GRAYSCALE or 0

cv2.IMREAD_COLOR or 1

The default value for flags is 1, which will read in the image as a Colored image.

For Example:

```
# Python code for reading an image
img_color = cv2.imread('test.jpg’, cv2.IMREAD_COLOR)
img_grayscale = cv2.imread('test.jpg’, cv2.IMREAD_GRAYSCALE)
img_unchanged = cv2.imread('test.jpg’, cv2.IMREAD_UNCHANGED)
```

Writing An Image File:

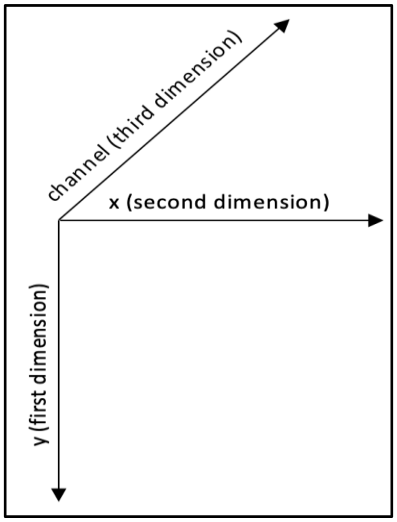
For reading an image, the imwrite() function is used in OpenCV. Here’s the syntax: imwrite(filename, image).1) The first argument is the filename, which must include the filename extension (for example .png, .jpg etc). OpenCV uses this filename extension to specify the format of the file. 2)The second argument is the image you want to save. The function returns True if the image is saved successfully

For Example:

```
cv2.imwrite('grayscale.jpg',img_grayscale)
```

Converting Between an Image and raw bytes

A byte is an integer ranging from to 255. Throughout real-time graphics applications today, a pixel is typically represented by one byte per channel, though other representations are also possible. An OpenCV image is a 2D or 3D array of type an 8-bit grayscale image is a 2D array containing byte values. A 24-bit BGR image is a 3D array, also containing byte values. We may access these values by using an expression like image[0, 0] or image[0, 0, The first index is the pixel's y coordinate, or row, being the top. The second index is the pixel's x coordinate, or column, being the leftmost. The third index (if applicable) represents a color channel. The array's three dimensions can be visualized in the following Cartesian coordinate system:



Accessing image data with numpy. Array

The numpy.array class enables some bulk manipulations that are not possible with a simple Python list because it is well optimized for array operations. These numpy.array type-specific methods are useful for OpenCV image manipulation.

Reading/Writing a video file

OpenCV provides the Video Capture and Video Writer classes, which support various video file formats. The supported formats vary depending on the operating system and the build configuration of OpenCV, but normally it is safe to assume that the AVI format is supported. Via its read method, a Video Capture object may

be polled for new frames until it reaches the end of its video file. Each frame is an image in a BGR format.

Conversely, an image may be passed to the write method of the Video Writer class, which appends the image to a file in Video Writer.

Capturing camera frames

A stream of camera frames is represented by a Video Capture object too. However, for a camera, we construct a Video Capture object by passing the camera's device index instead of a video's filename. Let's consider the following example, which captures 10 seconds of video from a camera and writes it to an AVI file.

Displaying images in a window

One of the most basic operations in OpenCV is displaying an image in a window. This can be done with the imshow() function. If you come from any other GUI framework background, you might think it sufficient to call imshow() to display an image. However, in OpenCV, the window is drawn (or re-drawn) only when you call another function, waitKey. The latter function pumps the window's event queue (allowing various events such as drawing to be handled), and it returns the keycode of any key that the user may have typed within a specified timeout. To some extent, this rudimentary design simplifies the task of developing demos that use video or webcam input; at least the developer has manual control over the capture and display of new frames.

For Example,

```
cv2.imshow('my image', img)
```

The imshow() function takes two parameters: the name of the window in which we want to display the image and the image itself.

Displaying camera frames in a window

OpenCV allows named windows to be created, redrawn, and destroyed using the namedWindow, imshow(), and destroyWindow functions. Also, any window may capture keyboard input via the waitKey function and mouse input via the setMouseCallback function. The argument for waitKey is a number of milliseconds to wait for keyboard input. By default, it is 0, which is a special value meaning infinity. The return value is either -1 (meaning that no key has been pressed) or an ASCII keycode, such as 27 for Esc.

Conclusion:

In this experiment, we have completed the basic ideas and procedures for utilizing OpenCV to manage files, cameras and graphical user interfaces (GUIs) for computer vision applications. The best way to handle various image and video sources is process them and show the outcomes in windows. I/O scripts are essential for dealing with photos and videos and covered various relevant functions.