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

Reading/Writing an Image File

Reading An Image File

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

imread(filename, flags)

It takes two arguments:

1. The first argument is the image name, which requires a fully qualified pathname to the file.
2. 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. When you want to read in an image in a particular format, just specify the appropriate flag.

Writing An Image File

Let's discuss how to write/save an image into the file directory, using the imwrite() function. Check out its 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.

Take a look at the code below. See how simple it is to write images to disk. Just specify the filename with its proper extension (with any desired path prepended). Include the variable name that contains the image data, and you're done.

Converting Between an Image and raw bytes

Conceptually, a byte is an integer ranging from 0 to 255. Throughout real-time graphic 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 the numpy.array type. An 8-bit grayscale image is a 2D array containing byte values. A 24-bit BGR image is a 3D array, which also contains byte values. We may access these values by using an expression such as image[0, 0] or image[0, 0, 0]. The first index is the pixel's *y* coordinate or row, 0 being the top. The second index is the pixel's *x* coordinate or column, 0 being the leftmost. The third index (if applicable) represents a color channel.

Accessing image data with numpy. Array

Reading/Writing a video file

Reading A Video File

In OpenCV, a video can be read either by using the feed from a camera connected to a computer or by reading a video file. The first step towards reading a video file is to create a VideoCapture object. Its argument can be either the device index or the name of the video file to be read.

In most cases, only one camera is connected to the system. So, all we do is pass '0' and OpenCV uses the only camera attached to the computer. When more than one camera is connected to the computer, we can select the second camera by passing '1', the third camera by passing '2' and so on.

Writing A Video File

Let's now take a look at how to write videos. Just like video reading, we can write videos originating from any source (a video file, an image sequence, or a webcam). To write a video file:

- Retrieve the image frame height and width, using the get() method.
- Initialize a video capture object (as discussed in the previous sections), to read the video stream into memory, using any of the sources previously described.
- Create a video writer object.
- Use the video writer object to save the video stream to disk.

Continuing with our running example, let’s start by using the get() method to obtain the video frame width and height.

Capturing camera frames

A frame is an image that forms a single instance of a video. A video consists of a lot of frames running per second (also known as frames per second). For example, a video streaming at 30 fps means that it displays 30 images in a second.

1. As we have to use OpenCV methods, they have to be included in our code file. Use the following line of code to include OpenCV library functions in Python `import cv2`.

2. The above code will include all the libraries defined in `cv2` library (OpenCV library in Python). Now, the next step is to create a variable that will hold the video capturing object. In this case, the variable is `vidcap`, which will hold the reference returned by the `VideoCapture` function of the `cv2` library.

3. The `vidcap` object created will now have the reference to the integrated webcam (for laptops) or any other portable webcam attached via input ports. The next step is to check if `cv2.VideoCapture` is successfully used. Use `vidcap.isOpened()`. It will return `true` if the connection with the camera was successful. Otherwise, `false` will be returned. If `true` , is returned, we will continue. Otherwise, we will print the error message.

4. If our connection with the camera was successful, we’ll capture a frame from the live video. For that, use the following line of code inside the `if` block from the previous step.

5. If the frame was successfully captured, we will display the captured frame by using cv2.imshow(windname, frame). The captured frame will be displayed in a new window. The windname argument is a string for the name you want to give to your frame that will be displayed with the frame window and the frame attribute is an object that holds the frame captured from the previous step.

Displaying images in a window

In OpenCV, you display an image using the imshow() function. Here's the syntax:

imshow(window_name, image)

This function also takes two arguments:

1. The first argument is the window name that will be displayed on the window.
2. The second argument is the image that you want to display.

To display multiple images at once, specify a new window name for every image you want to display.

The imshow() function is designed to be used along with the waitKey() and destroyAllWindows() / destroyWindow() functions.

The waitKey() function is a keyboard-binding function.

- It takes a single argument, which is the time (in milliseconds), for which the window will be displayed.
- If the user presses any key within this time period, the program continues.
- If 0 is passed, the program waits indefinitely for a keystroke.
- You can also set the function to detect specific keystrokes like the Q key or the ESC key on the keyboard, thereby telling more explicitly which key shall trigger which behavior.

Displaying camera frames in a window

OpenCV allows named windows to be created, redraw and destroyed using thenamedWindow(),imshow(), and destroywindow() functions . Also, any window may capture keyword input via waitKey() function and mouse input via the set MouseCallback() function.

Conclusion:

We have finally, grasped how a computer vision application can be developed openCV consists of mat class which is used for pixel manipulation and it has a class relationship to application buffered image class.