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

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short.Python offers various modules for creating GUI applications, out of which, Tkinter lets users do various tasks inside their app.

Reading/Writing an Image File

Python supports very powerful tools when it comes to image processing.Following are some libraries to process the images ImageIO, OpenCV, Matplotlib, PIL, etc .

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]

Accessing image data with numpy. Array

Images are an easier way to represent the working model. In Machine Learning, Python uses the image data in the format of Height, Width, Channel format. i.e. Images are converted into Numpy Array in Height, Width, Channel format.

Reading/Writing a video file

OpenCV (Open Source Computer Vision) is a computer vision library that contains various functions to perform operations on Images or videos. OpenCV library can be used to perform multiple operations on videos. To capture a video, we need to create a VideoCapture object. VideoCapture have the device index or the name of a video file. Device index is just the number to specify which camera. If we pass 0 then it is for first camera, 1 for second camera so on. We capture the video frame by frame.

Capturing camera frames

Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through a webcam and then you may perform desired operations on that video.

Steps to capture a video:

1. Use cv2.VideoCapture() to get a video capture object for the camera.

2. Set up an infinite while loop and use the read() method to read the frames using the above created object.
3. Use cv2.imshow() method to show the frames in the video.
4. Breaks the loop when the user clicks a specific key.

Displaying images in a window

PIL is the Python Imaging Library which provides the python interpreter with image editing capabilities. The Image module provides a class with the same name which is used to represent a PIL image. The module also provides a number of factory functions, including functions to load images from files, and to create new images.Image.show() Displays this image. This method is mainly intended for debugging purposes. On Unix platforms, this method saves the image to a temporary PPM file, and calls the xv utility. On Windows, it saves the image to a temporary BMP file, and uses the standard BMP display utility to show it (usually Paint).

Displaying camera frames in a window

Python libraries are independent and thus they all can be used for different purposes while building a particular featured application. In this example, we will build an application using OpenCV and Tkinter library. OpenCV is a Python library that is used to work with Computer Vision and other artificial artifacts. Using the OpenCV module, we have to show the webcam in a tkinter window.

Conclusion:

We have successfully completed this experiment, gaining practical knowledge and proficiency in performing basic I/O scripts for image and video files, converting between images and raw bytes, accessing image data with numpy arrays, capturing camera input, and displaying images and camera frames in windows. These foundational skills empower us to manipulate and analyze image data effectively, paving the way for more advanced and sophisticated machine vision applications.