Laboratory Exercise 2
OpenCV Videos

Submitted by:

Group # 7

SO	PI	Category	Exceptional 4	Acceptable 3	Marginal 2	Unacceptable 1	Score
b	1	**Compliance** 30%	All procedures were followed, the output is as expected, additional related functionalities were augmented.	All procedures were followed and the output is as expected.	All procedures were followed but the output is not as expected.	Did not follow the set procedures	
b	1	**Analysis** 20%	Data interpretation is professionally written with appropriate and clear illustrations.	Data is clearly and correctly explained with proper illustrations.	Data is not clearly explained, has minor flaws, or no illustrations	There is no data explanation about the data or results.	
b	1	**Validity** 20%	The implementation uses the concepts and principles of the experiment as well as advanced topics.	The implementation uses the concepts and principles of the theory discussed for the experiment.	Implementation did not clearly express the use of theory discussed for the experiment.	There is no implementation.	
b	1	**Interpretation** 20%	The conclusion is professionally written and points the theories in the experiment and its implications in engineering.	The conclusion points to the main ideas and applications of the theory in the experiment.	The conclusion does not point out the main ideas and applications of the theory in the experiment.	There is no conclusion.	
		Format and Clarity 10%	Follows the prescribed format, observes proper and technical grammar, and observes proper citation and referencing according to IEEE journal standards.	Follows the prescribed format, observes proper and technical grammar, and observes proper IEEE citation and referencing.	Did not follow the prescribed format, has poor grammar, or incorrect citations and references scheme.	Did not follow the prescribed format, has poor grammar, and has no citations and references.	
		TOTAL SCORE					

Group Members			
STUDENT NUMBER	**NAME**	**CONTRIBUTION**	**SCORE**
202113843	Betchayda, Ezekiel	Programming/Coding, Results and Discussion	
201911991	Marquita, Bhea Marrianne	Programming/Coding, Results and Discussion	
202111103	Mejia, Zaldy Enrico Louis	Programming/Coding, Results and Discussion, Conclusion	
202110017	Sangco, Jerrold Cornelius	Programming/Coding, Results and Discussion	

Submitted to:
Engr. Dexter James L. Cuaresma

Date:
02/12/2024

OBJECTIVES

- To be create a simple code to read video, display video and save video.
- To create a simple python program to video using camera and display it in pychram using OpenCV.

DISCUSSION

Capture Video from Camera

Often, we have to capture live stream with a camera. OpenCV provides a very simple interface to do this. Let's capture a video from the camera (I am using the built-in webcam on my laptop), convert it into grayscale video and display it. Just a simple task to get started.

To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. A device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So I simply pass 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture. (OpenCV, 2020)

Playing Video form File

Playing video from file is the same as capturing it from camera, just change the camera index to a video file name. Also while displaying the frame, use appropriate time for cv.waitKey(). If it is too less, video will be very fast and if it is too high, video will be slow (Well, that is how you can display videos in slow motion). 25 milliseconds will be OK in normal cases. (OpenCV, 2020)

Saving a Video

So we capture a video and process it frame-by-frame, and we want to save that video. For images, it is very simple: just use cv.imwrite(). Here, a little more work is required.

This time we create a VideoWriter object. We should specify the output file name (eg: output.avi). Then we should specify the FourCC code (details in next paragraph). Then number of frames per second (fps) and frame size should be passed. And the last one is the isColor flag. If it is True, the encoder expect color frame, otherwise it works with grayscale frame. (OpenCV, 2020)

FourCC is a 4-byte code used to specify the video codec. The list of available codes can be found in fourcc.org. It is platform dependent. The following codecs work fine for me.

In Fedora: DIVX, XVID, MJPG, X264, WMV1, WMV2. (XVID is more preferable. MJPG results in high size video. X264 gives very small size video)

In Windows: DIVX (More to be tested and added)
In OSX: MJPG (.mp4), DIVX (.avi), X264 (.mkv).
FourCC code is passed as `cv.VideoWriter_fourcc('M','J','P','G')or cv.VideoWriter_fourcc(*'MJPG')` for MJPG. (OpenCV, 2020)

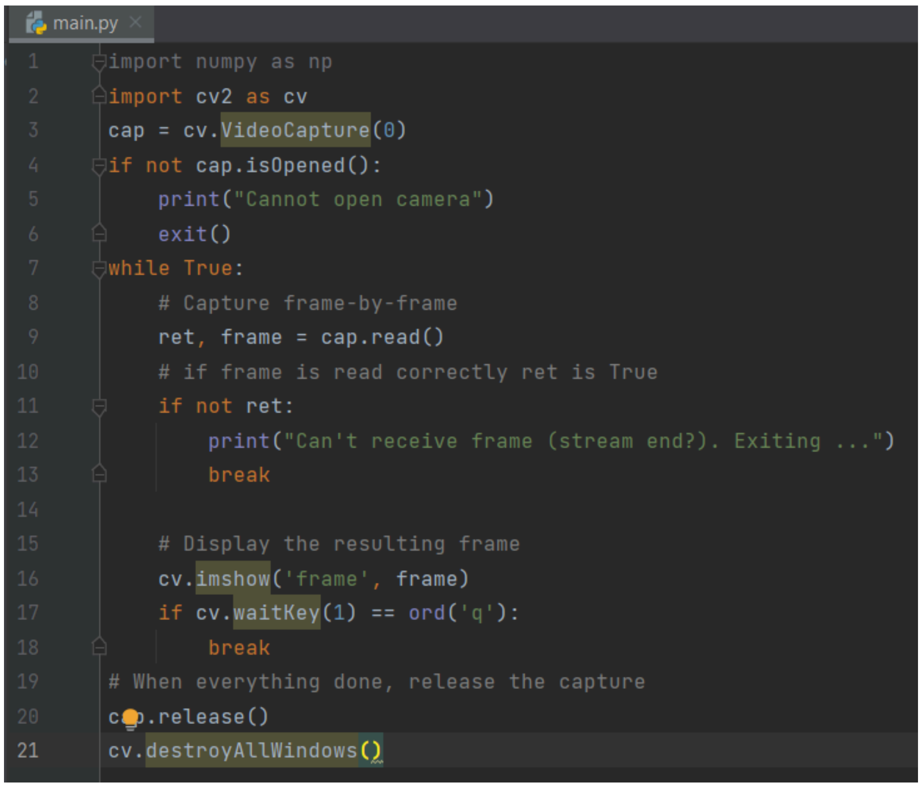
MATERIALS

Software:

- PyCharm
- OpenCv
- Anaconda
- Python

PROCEDURES

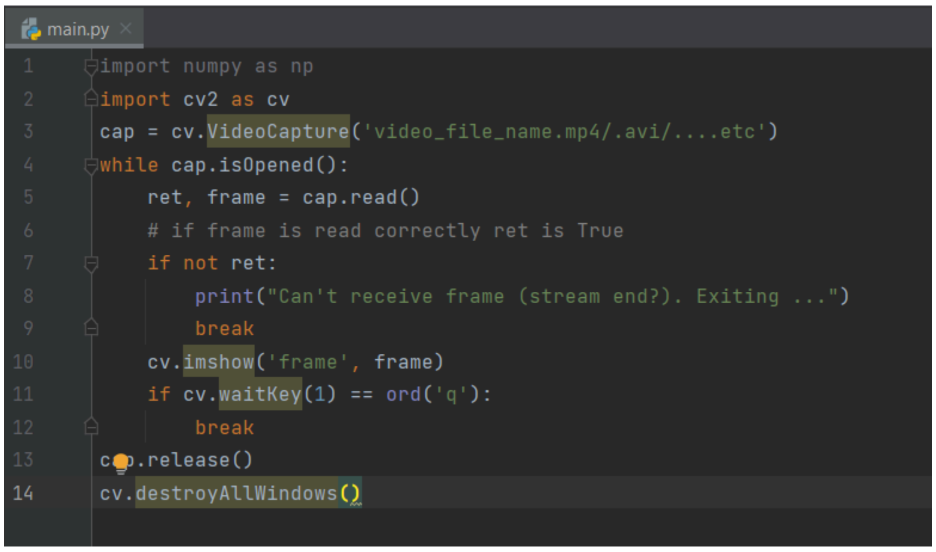
1. Run the given code below and write your observation
   a. ***Capture Video From Camera***



```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    # Display the resulting frame
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv.destroyAllWindows()
```

   b. ***Playing Video from File***

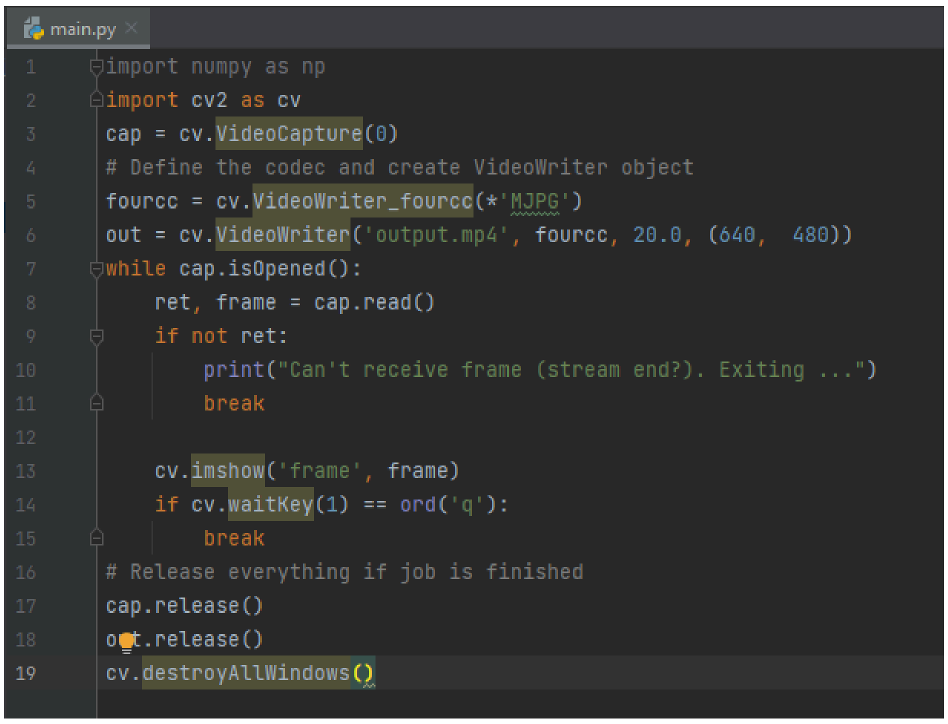
   Note: upload your video file in the directory folder



```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture('video_file_name.mp4/.avi/....etc')
while cap.isOpened():
    ret, frame = cap.read()
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

c. ***Saving a Video***

Note: You can run your created video file by using the code in letter b to check your output.



main.py

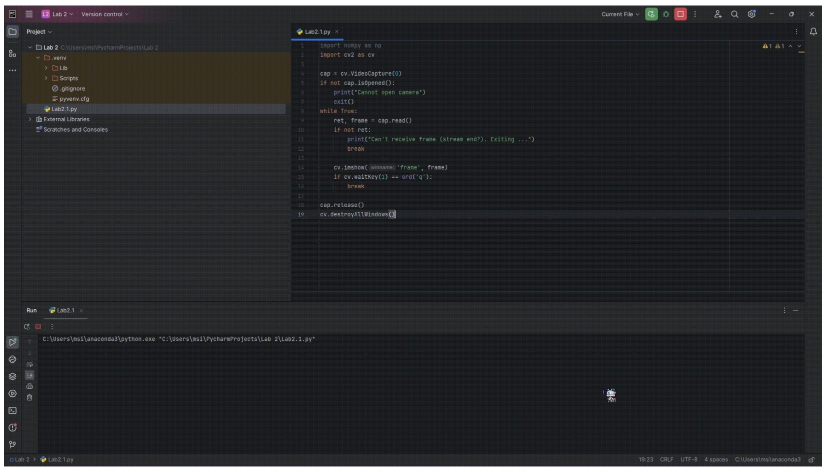
```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
# Define the codec and create VideoWriter object
fourcc = cv.VideoWriter_fourcc(*'MJPG')
out = cv.VideoWriter('output.mp4', fourcc, 20.0, (640,  480))
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break
# Release everything if job is finished
cap.release()
out.release()
cv.destroyAllWindows()
```

2. Take a screenshot of your design and **describe each image**.
3. Save your works as Lab2_GroupNo. as a PDF file. And upload it to the submission file.

RESULTS AND DISCUSSION

❖ OpenCV:

a. Example 1



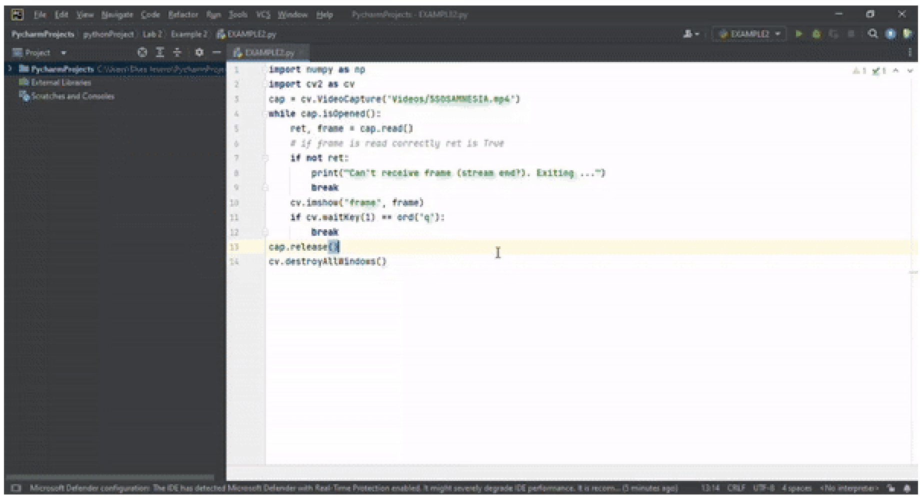
```python
import numpy as np
import cv2 as cv

cap = cv.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    cv.imshow( winname: 'frame', frame)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

```
C:\Users\msi\anaconda3\python.exe "C:\Users\msi\PycharmProjects\Lab 2\Lab2.1.py"
```

Figure 1. Demonstration of VideoCapture()

As shown in Figure 1, by using OpenCV's VideoCapture function to open the camera. The code line "if not cap.isOpened():" works if there is a camera that is open or connected to the computer, and if no camera is detected it will print "Cannot open camera". The code line "while True: ret, frame = cap.read()" works if the frame is readable. If not, then it means that there is a problem in the software of the camera. Otherwise, if the camera is working properly it will display the resulting frame of the camera with the use of the code line "cv.imshow("frame", frame)". Additionally, the code line "if cv.waitKey(1000) == ord('q'):" adds a function to close the window once the assigned key is pressed. Lastly, the code lines "cap.release()" and "cv.destroyAllWindows()" functions to release the resource that was initially called in the code, which means that if the VideoCapture is using the camera, no other process system can use the camera. The other code line functions to simply destroy all windows that were created in the code.

b. Example 2



```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture('Videos/SSDSAMNESIA.mp4')
while cap.isOpened():
    ret, frame = cap.read()
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

Figure 2. Demonstration of loading local videos using VideoCapture()

As shown in Figure 2, cap is set to open a local video file using OpenCV’s VideoCapture() function. ret is a boolean variable that returns true if the frame is available. So if the frame is not available, the program will print a message that says “Can’t receive frame…” and then it will automatically end process. If the frame is available, the video will play and the program will wait for the user to click q using the waitKey() function. If the letter q on the keyboard is clicked, the program will destroy all windows via the destroyAllWindows() function. Otherwise, the frame will not close.

c. Example 3

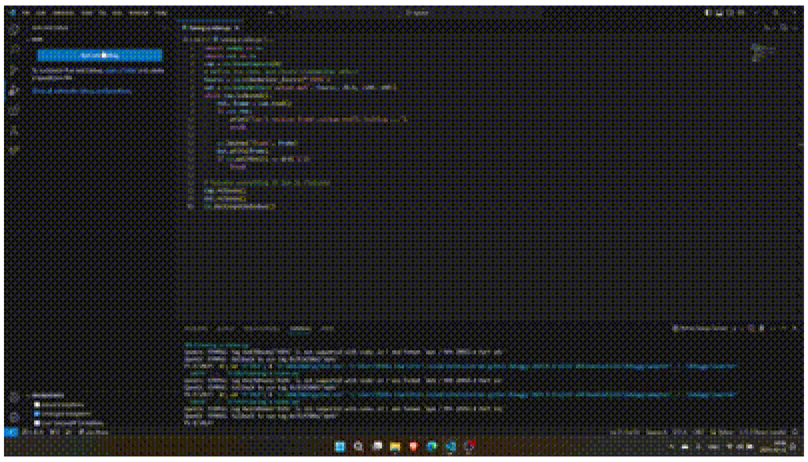


Figure 3. Demonstration of VideoWriter()

Similar to the first example, the learners have used OpenCV's videocapture function to capture video footage from a camera connected to the device, in this case, camera 0. to specify a codec used in the programme, the function used is fourcc = cv.VideoWriter_fourcc(*'MJPG'), this is used to ensure that the video captured has been compressed into a format readable in the OS, using different codec formats such as mp4v or XVID and DIVX can also used for mp4 or mpeg4 files, without inserting proper codecs that are compatible will yield to a corrupted file due to it being incompatible. The object is from the function VideoWriter with attributes following this format (Filename, codec, framerate, and screen size). The rest follow the same principle as example 1, the only deviation of this is having a function out.write() in which it tells the VideoWriter to write the recorded data, out.release() in which it tells VideoWriter to close.

Github link of the repository for the experiment:
https://github.com/bheanne/MLPGrp7/tree/abc987f7bc82d3afd0a811a64d37deff349f6570/LabAct2

CONCLUSION

In conclusion, the learners were introduced to the new functions/classes of Opencv through code examples which allowed the learners to open the camera, play a video from their local PCs, and record a By using the cv2.VideoCapture() class, they were able to access and utilize camera devices connected to their computers, allowing them to capture real-time video streams. By changing the arguments of the cv2.VideoCapture() class, the learners learned to load and play video files stored on their PCs by coding the directory path of the video of their choosing. Furthermore, they explored the process of recording video streams directly to a file on their computers with the cv2.VideoWriter() class. It allowed them to explore various codecs and media files and to choose which codec is suited for what type of media file. Overall, the coding examples given expanded the learner's skill set in openCV and prepared them for more advanced experiments/coding examples.