# **Assignment 1 – Group 5** - Mohammed Aadil, Jonathan Chacko, Aliyyah Jackhan

# **Introduction to OpenCV, Capturing Videos, Digital Images & Color Standards**

| Total Mark: | 12 marks (6% of the total course grade)   * 9 out of 12: Learn@Seneca Submission (Due: Monday May 26 at 8:00am) * 3 out of 12: Assignment Demo (During the Lab of Week 4) |
| --- | --- |
| Submission file(s): | * Assignment1\_1.py / Assignment1\_1.ipynb * Assignment1\_2.py / Assignment1\_2.ipynb * Assignment1.docx (this document with your answers) |

Please work **within your group** to complete this assignment.

This assignment is worth 6% of the total course grade and will be evaluated through your written submission, as well as the assignment demo.

During the assignment demo, group members are *randomly* selected to explain the submitted solution. Group members who are not present during the assignment demo will lose the demo mark.

Please submit the submission file(s) through Learn@Seneca.

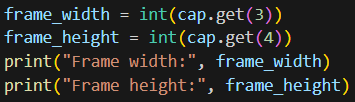
***Please paste the resulting images and answers in this document.***

## **Part I: A Simple OpenCV Project**

For this assignment, you need a webcam, or a digital camera connected and installed on your machine.

1. Open **Anaconda Navigator**, then on Home tab, choose **ocv/socv** environment.
2. Launch PyCharm, Visual Studio Code, or Jupyter Notebook (whichever you prefer) and select ocv as the virtual environment.
3. Create a program (save as Assignment01\_1). Copy and paste the following code in a code block (Reference: [OpenCV: Getting Started with Videos](https://docs.opencv.org/4.11.0/dd/d43/tutorial_py_video_display.html)).

|  |
| --- |
| import cv2 as cv  # Start a video capture, using device's camera  cap = cv.VideoCapture(0)  # Check if video file opened successfully  if (cap.isOpened() == False):      print("Error opening video stream or file")  frame\_width = int(cap.get(3))  frame\_height = int(cap.get(4))  print("Frame width: " , frame\_width)  print("Frame height: " , frame\_height)  # Read until video is completed  while(cap.isOpened()):      # Capture frame-by-frame      ret, frame = cap.read()      if ret == False:          break      # Display the frame      cv.imshow('frame',frame)      key = cv.waitKey(25)      # Press Q on keyboard to exit      if key & 0xFF == ord('q'):          break  # Release the video capture  cap.release()  # Close all the frames  cv.destroyAllWindows() |

1. Run the code. You should see your webcam’s video feed. Press ‘q’ to exit.
2. Briefly explain what the code is doing.
   1. Captures video from the device’s webcam.
   2. Displays each video frame in a window titled *‘frame’.*
   3. Press *‘q’* to exit.
   4. Uses *‘cv.waitKey(500)’* to set frame delay.
   5. Ends the stream and closes all windows when done.
3. What are the frame width and height values?
   1. These values vary based on your camera. The program prints them using:  
        
        
      Example Output:   
      
4. Change the parameter for cv.waitKey() from 25 to 100, and 500. What happens? Explain.
   1. *Cv.waitKey(25):* Video appears fast and smooth.
   2. *Cv.waitKey(100):* Frame display slows down, slightly choppy.
   3. *Cv.waitKey(500):* Frame display significantly delayed, very slow, almost like taking snapshots.
   4. Why does this difference occur: *Cv.waitKey()* determines how long (in milliseconds) OpenCV waits for a key event. It indirectly controls the FPS — higher values lower the FPS and slow down the video feed.

## **Part II: A Photo Booth Application (Read, Display, Write, and Pad Images)**

Modify Assignment1\_1 to:

1. Take a snapshot whenever the ‘x’ key is pressed.
2. Crop 15 pixels around the snapshot image.
3. Pad the snapshot with 40 pixels using replicate padding (or a constant boarder with the color you like) See: [OpenCV: Adding borders to your images](https://docs.opencv.org/4.11.0/dc/da3/tutorial_copyMakeBorder.html).
4. Save the snapshot. Use image names such as ‘image01.jpg’, ‘image02.jpg’, etc., automatically incrementing the filename counter. See: [OpenCV: Getting Started with Images](https://docs.opencv.org/4.11.0/db/deb/tutorial_display_image.html).
5. Show this image in a new window for 1 second. Then automatically close it and return to the camera feed.
6. Exit whenever the ‘q’ or ESC key is pressed.

Paste one of the snapshots here.



## **Part III: Calculations**

For this part, just add your answers in the document.

Calculating FPS:

1. Calculate the approximate FPS implemented in the code (use *waitKey()*). Ignore all other delays including 1 second freezes.
   1. If *cv.waitKey(500)* is used, the FPS ≈ 1000 / 500 = **2 FPS**.
2. How would you change the code to have an FPS of 5? (once use *waitKey()* and once use a method with *cap*).
   1. Using waitKey():
      1. Code: *cvwaitkey(200) #1000 ms / 5 FPS = 200 ms delay*
   2. Using cap.set() method:
      1. Code: *cap.set(cv.CAP\_PROP\_FPS, 5)*
3. What do you observe when you apply each solution? What are the limitations of each method?
   1. Observation:
      1. *waitKey()* controls display rate but not actual camera capture FPS.
      2. *cap.set()* attempts to control camera FPS but might be ignored by some webcams.
   2. Limitation:
      1. Not all cameras support changing FPS via *cap.set()*; it's hardware-dependent.
4. Find image01.jpg and look at its properties.
5. What is the image resolution (dimensions)?
   1. 720 × 600 pixels
6. Does this match the width and height output of Part I? Why is that?
   1. No. The original camera resolution (likely 640×480) was changed due to:
      1. 40 pixels of padding added to each side → adds 80 to width and height.
      2. Any crop adjustments might also slightly alter the height.
7. What is the bit depth? What does bit depth show?
   1. Bit depth: 24 bits (8 bits per channel × 3 channels for RGB)
   2. Bit depth represents the number of bits used for each pixel’s color data.
   3. Higher bit depth = more color detail.
8. What is the file size in bytes?
   1. 75,076 bytes (≈ 75 KB)
9. What would the file size be if the image was not compressed? Show the calculations to support.
   1. Uncompressed size: Width × Height × Channels = 720 × 600 × 3 = 1,296,000 bytes ≈ 1.3 MB
10. Calculate the compression ratio as the ratio between the uncompressed size and the compressed size.
    1. Uncompressed size / Compressed size = 1,296,000 / 75,076 ≈ 17.26;   
       So, compressed ratio ≈ 17.3:1
11. How do you know that image is compressed? Explain.
    1. The image is in JPEG format, which uses lossy compression.
    2. The actual file size is much smaller than the uncompressed size.
    3. JPEG compression introduces efficiency by discarding non-critical visual data.

## **Part IV: Color Standards**

Create a program (save as Assignment01\_2). Include code to:

Open a color image and display. Paste a sample here.



1. Convert it from BGR to HSV color-space (See: [OpenCV: Color Space Conversions](https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab)).
2. Set the H once to 0 and once to 90. Paste the results here.

H = 0 image   


H = 90 image  


1. What is being done in this part? Explain.
   1. In this part, the image is being processed using HSV color space instead of the default BGR (Blue-Green-Red) format used by OpenCV.
   2. The key steps are:
      1. Convert BGR to HSV:
         1. The original image is converted from BGR to HSV (Hue, Saturation, Value) color space using cv.cvtColor().
         2. This conversion separates color information (Hue) from brightness (Value), which makes color manipulation easier and more intuitive.
   3. Modify the Hue channel:
      1. The Hue (H) component of the image is manually set to specific values:
         1. H = 0 → all pixels shift toward red hues.
         2. H = 90 → all pixels shift toward green hues.
      2. Saturation and Value channels are kept the same.
   4. Convert HSV back to BGR and display/save:
      1. The modified HSV image is converted back to BGR so it can be displayed and saved properly using OpenCV.

Please note that, for HSV color-space in OpenCV, hue range is [0,179], saturation range is [0,255], and value range is [0,255].

## **Part V: Group Work**

Add this declaration to your file:

We, Group 5 – Mohammed Aadil, Jonathan Chacko, Aliyyah Jackhan, declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.

Specify what each member has done towards the completion of this assignment:

|  |  |  |
| --- | --- | --- |
|  | Name | Task(s) |
| 1 | Mohammed Aadil | Formatting the final doc and completing the part 4 of the assignment. |
| 2 | Jonathan Chacko | Completed the part 1 and part 2 of the assignment. |
| 3 | Aliyyah Jackhan | Completed the part 3 of the assignment. |