WorkShop4

Contents

WorkShop4

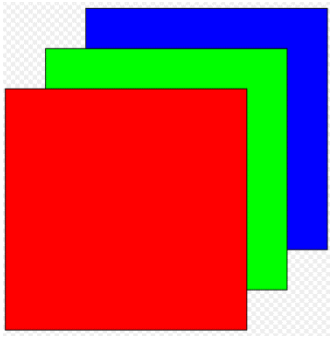
Instructions:

- The workshop can be completed **in groups (maximum of four members per group (recommended))**.
- All group members should work together, and they will receive the same mark.
- This workshop is worth 2.5% of the total course grade, and it will be graded out of 25 marks and evaluated through your written submission, as well as the lab demo as follows:
    - 25 marks (2.5% of the total course grade)
        - 15 out of 25 (60%): Blackboard submission
        - 10 out of 25 (40%): Lab demo during the lab session
- Please submit the submission file(s) through Blackboard. **Only one person must submit for the group; only the last submission will be graded**.
- During the lab demo, group members are ***randomly*** selected to explain the submitted solution.
- **Group members who do not present during the lab demo will lose the demo mark.**

WorkShop4

Part One: Description

This workshop is just to make sure you can read an image, display an image, view an image in the different color models (**RGB** and **HSV**), and manipulate an image by splitting color channels and merging them again. Also, you will produce some output images that you will submit along with your code and the written report.



Remember that a digital image is a Two-Dimensional signal described by the brightness or color of picture elements ("pixels") indexed by horizontal and vertical coordinates. By default, a color image is stored by OpenCV-Python using 3 matrices, each representing **red**, **green** and **blue** components of pixels. Image/video processing is also referred to as R/G/B channels. A matrix is an array indexed by two indexing variables, typically for a row and a column. Note that the order of channels in OpenCV-Python is **BGR**. So, the first channel is blue, then green, followed by red channel.

OpenCV allows us to open an image and store it in a 3-dimensional array or matrix where the x and y-axis designate the pixel's location in the image, and the z-axis designates the RGB color channel. Each RGB pixel contains an **8-bit red component**, an **8-bit green component**, and an **8-bit blue component**. You can read images into a NumPy array using the OpenCV-Python library. The array contains pixel-level data. As per the requirement, you may modify the image data at a pixel level by updating the array values.

RGB representation of an image does not always offer complete insights into how color is seen by human eyes. For example, we may feel two things are of the same color hues, but one is lighter and more diluted while the other is more solid. HSV colour model is an alternative colour representation that offers such perceptual insights by giving numerical descriptions of **hue**, **saturation**, and **value** of brightness. HSV is useful in the artistic selection of colors and in identifying regions of interest in an image (e.g., locating people in an image by detecting skin tones).

WorkShop4

Numpy Library (Necessary because OpenCV uses it in the background)

Numpy help:

a) Python Numpy Tutorial (with Jupyter and Colab) : https://cs231n.github.io/python-numpy-tutorial/

b) Learn NUMPY in 5 minutes - BEST Python Library! : https://youtu.be/xECXZ3tyONo

c) UCSB Numpy Tutorial: https://sites.engineering.ucsb.edu/~shell/che210d/numpy.pdf

d) Numpy Tutorial: A Simple Example-based Guide: https://stackabuse.com/numpy-tutorial-a-simple-example-based-guide/

WorkShop4

Part Two: Workshop Structure

Download the starter code "ws4.zip" for workshop3 from Blackboard

The workshop folder/directory structure is as follows:

- **code**/: directory contains files named as **question<x>.py**. For example, **question1.py**, …. you need to write your code in these python files.
- **data**/: directory contains the input images, videos or other data supplied with this workshop.
- **output**/: directory must be used to store your output from this workshop. Your output can be images, videos, or other file types.
- **zip_submission.py**: This Python file will be used by you to create a zip file containing your code and output. To generate the submission once finished, use the following command: "**python zip_submission.py**".
    - Note: **run this command inside the ws4 directory**
    - You must submit this Zip file with your pdf report through Blackboard.
- ***.py:** add any other supporting files you may need to complete your workshop to the **code** directory.

WorkShop4

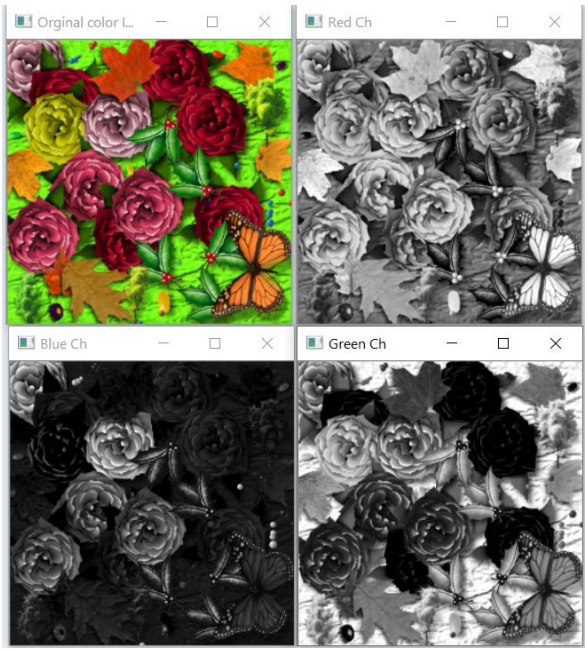
Part Three: Read an Image and display it in RGB/BGR Color Mode

Question 1. Write a code in **question1.py** file to do the following tasks:
Note: You can divide your code into functions.

Step1. Read the image "**Flower.bmp**" from the data folder with option **cv2.IMREAD_COLOR**
Step2. View the image on screen using **cv2.imshow**.
Step3. Now you have a color image from (**Step1**) that contains three channels/components for BGR. **Split** image channels and **view** them on screen separately (i.e., each channel in one window using **cv2.imshow**), then save each channel as an image inside the output folder with following names: "**<groupNO>_<questionNo>_FlowerBlueCh.jpg**", "**<groupNO>_<questionNo>_FlowerGreenCh.jpg**" and "**<groupNO>_<questionNo>_FlowerRedCh.jpg**", respectively.
For Example, Group1 save image as "1_q1_ FlowerRedCh.jpg"

Note: Each channel, **Blue**, **Green** or **Red**, is represented by default using 8 bits (unsigned 8-bit integer or '**uint8**' in NumPy array data type. So, 8-bit representation gives a scale of 0 to 255, with 0 being the **darkest** and 255 the **brightness** in the color channel.

>>>>>View the original image and different channels here<<<<<



WorkShop4

Step4. Write a short paragraph explaining what you see in the above images that you generated and saved in previous steps. Briefly discuss the procedure that you used to generate the above images.

>>>>>Write your answer here <<<<<

Answer the following sub-questions:

Sub-Question 1. How do the individual B, G, and R channels look when viewed separately? Are there any notable differences in brightness or contrast between the channels?

>>>>>Write your answer here <<<<<

Sub-Question 2. What is the significance of using an 8-bit representation for each channel? How does this affect the range of values each pixel can take?

>>>>>Write your answer here <<<<<

Sub-Question 3. Did you encounter any challenges or issues while splitting the channels or saving the images? How did you resolve them?

>>>>>Write your answer here <<<<<

WorkShop4

Step5. Step5: Now, we will show the results of adding color channels/components. Use **numpy.zeros** or **numpy.ones** to create an n-dimensional array (i.e., a new image) with an identical shape (i.e., dimensions) as an image in **Step1**.

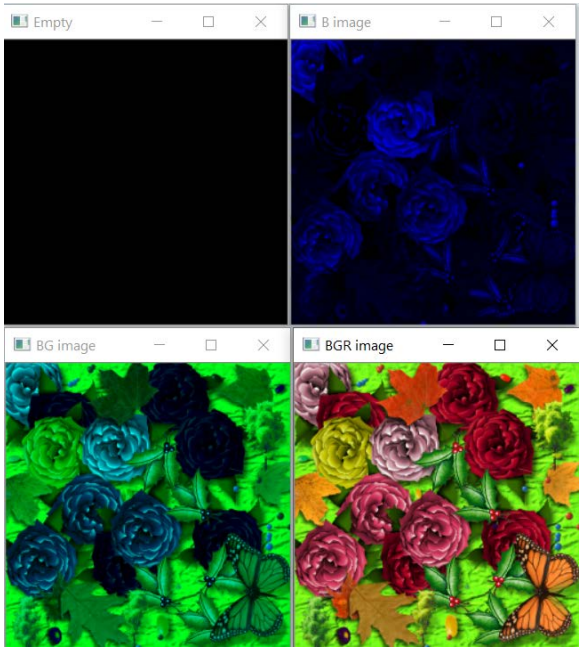
Note: you can read more about **numpy.zeros** from this link API Interface or **numpy.ones** from this link API Interface.

Step6. Add a **blue** channel from **Step3** to the newly constructed image in **Step5**, and then view the resulting image and save it inside the **output** folder with the name "**<groupNO>_<questionNo>_bimage.jpg**". **For Example, Group1 save image as "1_q1_bimage.jpg"**

Step7. Add the **green** channel from **Step3** to the resulting image from **Step6**, and then view the resulting image and save it inside the **output** folder with the name "**<groupNO>_<questionNo>_bgimage.jpg**".

Step8. Add a **Red** channel from **Step2** to the resulting image from **Step7**, and then view the resulting image and save it inside the **output** folder with the name "**<groupNO>_<questionNo>_bgrimage.jpg**".

>>>>>View the images from Step5, 6, 7, and 8 here<<<<<



WorkShop4

Step9. Write a small paragraph explaining what you see in the above images you generated and saved in **Step5, 6, 7, and 8**. Briefly discuss the procedure that you used to generate the above images.

>>>>>Write your answer here <<<<<

Answer the following sub-questions:

Sub-Question 4. Why is it important to create a new image array with the same shape as the original image?

>>>>>Write your answer here <<<<<

Sub-Question 5. Describe adding a single-color channel (e.g., blue) to the new image array. What changes did you observe in the image? How did adding the green channel affect the image created in the previous step?

>>>>>Write your answer here <<<<<

Step10. Repeat Question1 for **"falltreenature.jpg". Use the following name convention to save images inside the output folder "<groupNO>_<questionNo>_tree_XYZ.jpg".**

WorkShop4

Part Four: Read an Image and display it in HSV Color Mode

Question 2. Write a code in **question2.py** file to do the following tasks:
Note: You can divide your code into functions.

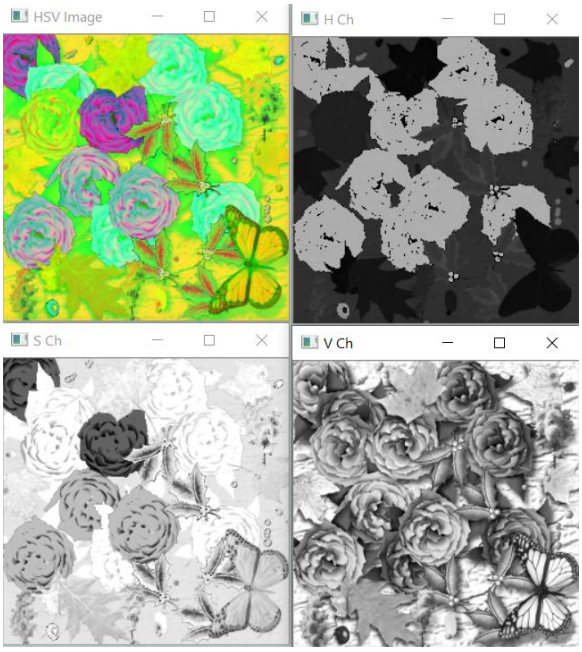
Step1. Read the image "**Flower.bmp**" from the data folder with option **cv2.IMREAD_COLOR**

Step2. View the image on screen using **cv2.imshow**.

Step3. Now you have a color image from (**Step1**) that contains three channels/components for BGR. Use OpenCV built-in function to convert the image from the BGR color model to the HSV model, and then view the resulting image and save it inside the output folder under the name "**FlowerHSV.jpg**".

Step4. Now **split** the image from **Step3** into different channels (**Hue**, **Saturation**, and **Value**) and **view** them on screen separately (i.e., each channel in one window using **cv2.imshow**), then save each channel as an image inside the output folder with following names: "**<groupNO>_<questionNo>_FlowerHCh.jpg**", "**<groupNO>_<questionNo>_FlowerSCh.jpg**" and "**<groupNO>_<questionNo>_FlowerVCh.jpg**", respectively. **For Example, Group1 save image as "1_q2_ FlowerVCh.jpg"**

>>>>>View the original image and different channels here<<<<<



WorkShop4

Step5. Write a paragraph explaining what you see in the above images you generated and saved in **Step4**. Briefly discuss the procedure that you used to generate the above images.

>>>>>Write your answer here <<<<<

Answer the following sub-questions:

Sub-Question 6. What are the characteristics of the Hue, Saturation, and Value channels? How do they differ from each other in terms of image representation?

>>>>>Write your answer here <<<<<

Sub-Question 7. In what practical situations might you need to convert an image from BGR to an HSV color model?

>>>>>Write your answer here <<<<<

WorkShop4

Step6. Now, we will show the results of adding color channels/components. Use numpy.zeros or numpy.ones create an n-dimension array (i.e., new image) with an identical shape (i.e., dimensions) as an image in **Step3**.

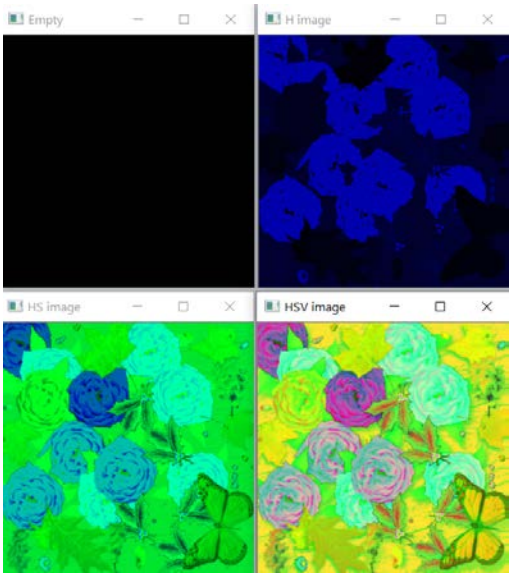
Note: you can read more about **numpy.zeros** from this link API Interface or **numpy.ones** from this link API Interface.

Step7. Add the **Hue** channel from **Step4** to the newly constructed image in **Step6**, and then view the resulting image and save it inside the output folder with the name "**<groupNO>_<questionNo>_himage.jpg**".

Step8. Add the **Saturation** channel from **Step4** to the resulting image from **Step7**, and then view the resulting image and save it inside the output folder with the name "**<groupNO>_<questionNo>_hsimage.jpg**".

Step9. Add the **Value** channel from **Step4** to the resulting image from **Step8**, and then view the resulting image and save it inside the output folder with the name "**<groupNO>_<questionNo>_hsvimage.jpg**".

>>>>>View the images from Step6, 7, 8, and 9 here<<<<<



WorkShop4

Step10. Write a small paragraph explaining what you see in the above images you generated and saved in **Step6, 7, 8, and 9**. Briefly discuss the procedure that you used to generate the above images.

>>>>>Write your answer here <<<<<

Step11. Repeat Question2 for **"falltreenature.jpg". Use the following name convention to save images inside the output folder "<groupNO>_<questionNo>_tree_XYZ.jpg".**

WorkShop4

Deliverables and Group Work

Create workshop report with the following name format

group_<number>_ws_<workshop number>_report.pdf

For example, if the **group16** created a report for **workshop20,** then the report name should be

group_16_ws_20_report.pdf

The workshop report should include:

(a) Complete this declaration by adding your names:

We, ---------- (mention your names), 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.

(b) Specify what each member has done towards the completion of this work:

	Name	Task(s)
1		
2		
3		
4		

(c) The report shows all your answers and outputs for all the workshop questions. So, you include the output images under the question (or part of question) and write response to answer some of the workshop questions.

(d) Submit two files

a. **submission.zip**
b. **group_<number>_ws_<workshop number>_report.pdf**