# Assignment 0 - Pixel Manipulation of Images and Videos

Name Vinit Mehta
Roll Number 2022111001

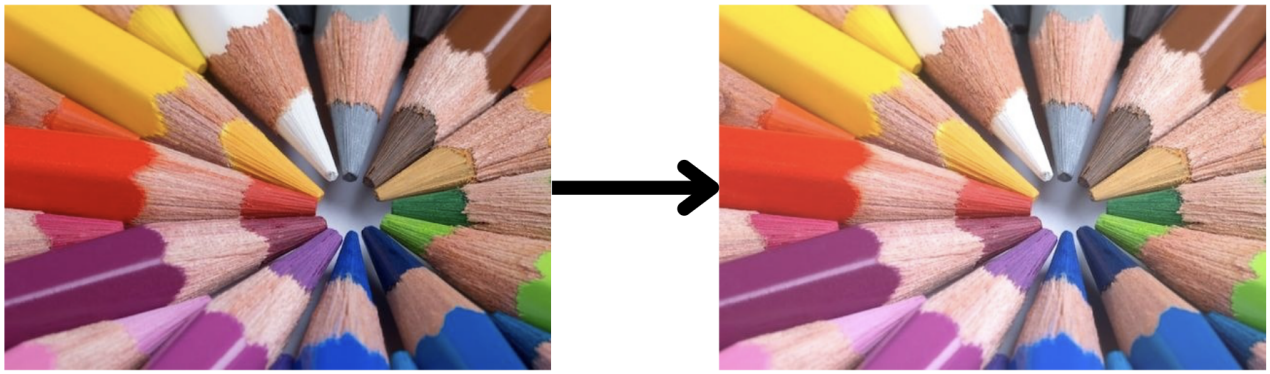
## Reading and Writing of Image

1. Reading the image from file into a numpy array.
2. Converting numpy array to python integer (32 bit) list.
3. While writing converting the integer list back to numpy array with datatype int8 and writing it onto a file.
4. Before writing we check if it's grayscale or not using a helper function which checks the number of channels per pixel to determine if the image is grayscale or not.

> Note
>
> OpenCV (cv2) reads the image in BGR format, so for specific purpose you can convert it to RGB format just by reversing the channel values for each of the individual pixels. And before writing back we are converting RGB back to BRG by calling the same function abstracted by another function.

## Changing Brightness

1. Brightness of image is depicted by the value of it's RGB channels for each pixel.
2. To increase (or decrease) the brightness of any pixel you can correspondingly increase (or decrease) the value of each channel by adding a fixed amount to each channel of each pixel.
3. This I have achieved through looping over the pixels and channels and adding a specific value to each.



> Note
>
> The pixel channel values are bounded by 0 and 255, so we clip the values if they exceeds these limits.

## Changing Contrast

**Contrast Enhancement by multiplication:** Loop throught the image array and for each pixel multiply some constant $\alpha$ in each of the pixel after centering it at 128 (in case of colored image multiply with the same factor in each channel individually) and decenter to again followed by finally clip the values to obtain the final image.

Contrast Enhancement by Multiplication

**Contrast Stretching:** The range of values that pixels in a picture have may not be covering the whole range from 0 to 255. So to enhance the contrast (distinction between areas) we can stretch this distribution so as to fit the whole range of values from 0 to 255.

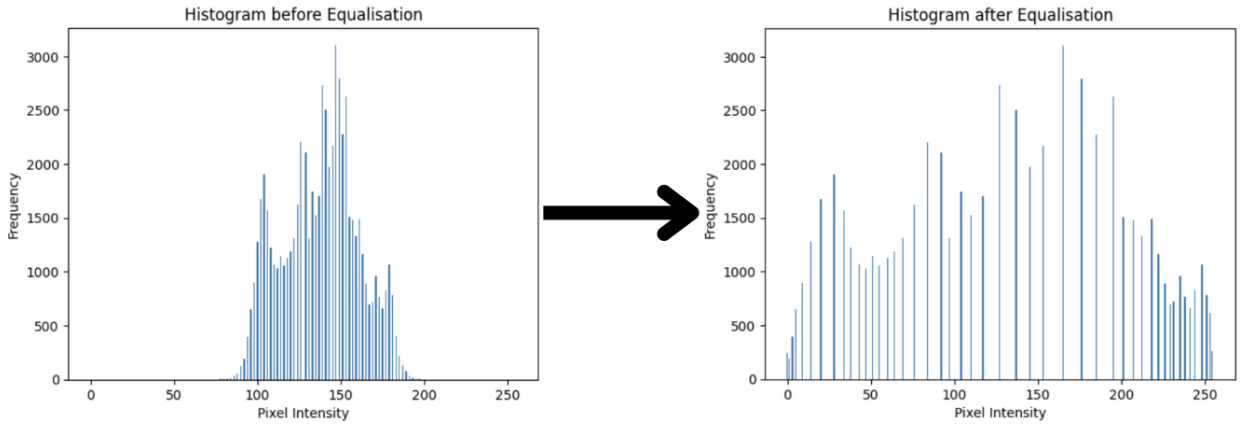
Contrast Stretching

This has been done using the following formula:

$$f_{ac}(a) = a_{min} + (a - a_{low}).\frac{a_{max} - a_{min}}{a_{high} - a_{low}}$$

**Histogram Equalisation:** The range of values that pixels in a picture have may not be distributed equally in range from 0 to 255. So to enhance the contrast (distinction between areas) we can equalise the distribution.

Histogram Equalisation




This has been done using the following formula:

$$p_r(r_k) = \frac{n_k}{MN}$$

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

$$k = 0, 1, 2, ..., L-1$$

# Color to Grayscale

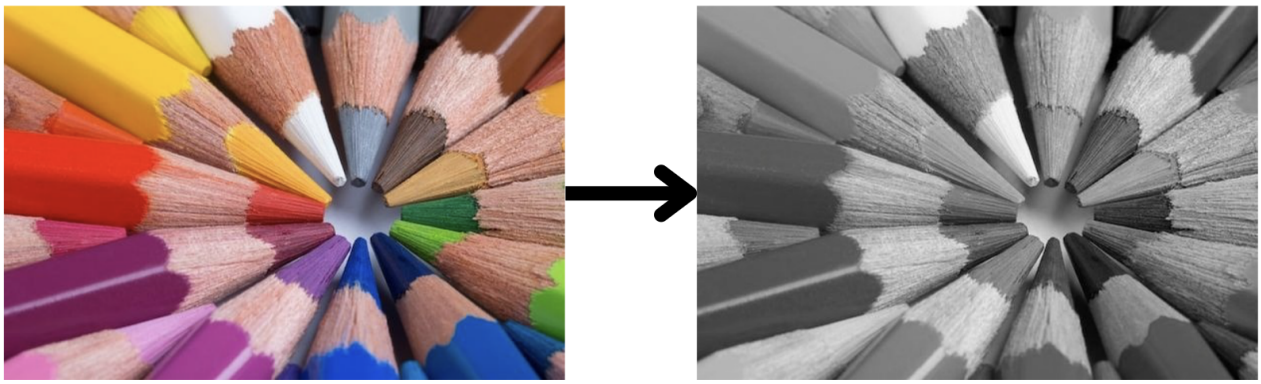
## Different Ways to Convert an Image to Grayscale and Their Effects

### 1. Average Method

- **Method:** Calculate the average of the red, green, and blue (RGB) color values for each pixel.
- **Formula:**

$$Gray = \frac{R+G+B}{3}$$

- **Effect:** This method gives equal weight to all three color channels, which may result in a dull or less contrasty grayscale image, especially if the image has strong color contrasts.



### 2. Luminosity Method (Weighted Average)

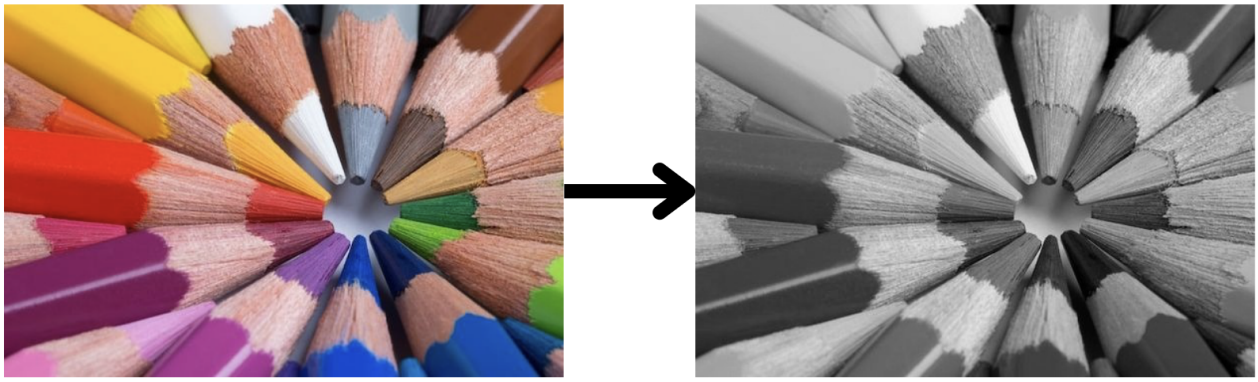
- **Method:** Apply different weights to the RGB channels, based on human perception of colors.
- Looped through each pixel and set R = G = B = value given by the below formula:

$$I'(x,y) = \alpha I_R(x,y) + \beta I_G(x,y) + \gamma I_B(x,y)$$

- For example in the code I have used:

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

- **Effect:** This method produces a grayscale image that more accurately reflects human perception, as the green channel is given more weight, and the blue channel is given the least. The result is typically a more natural and contrast-rich grayscale image.



Note

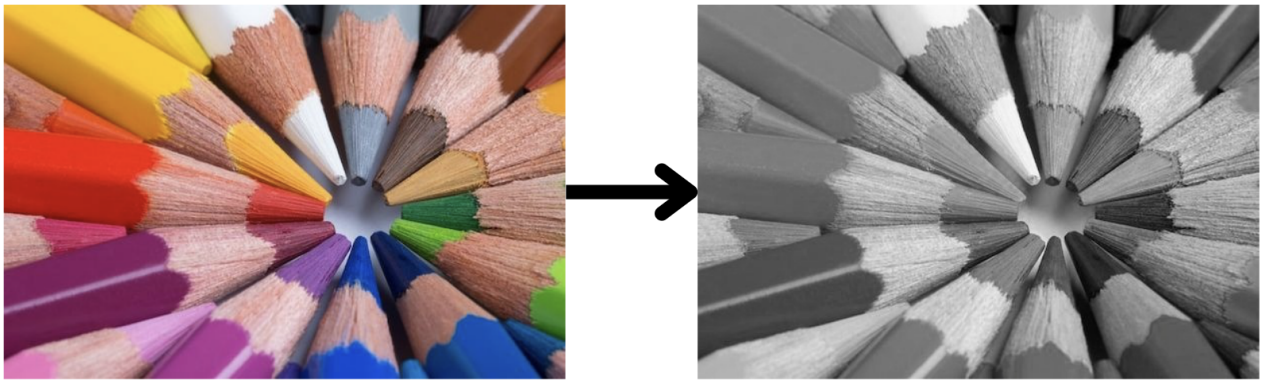
$\alpha + \beta + \gamma = 1$

**3. Desaturation Method**

- **Method:** Take the average of the maximum and minimum values of the RGB channels.
- **Formula:**

$$Gray = \frac{max(R, G, B) + min(R, G, B)}{2}$$

- **Effect:** This method results in a grayscale image that tends to preserve more detail in both highlights and shadows but might lose contrast compared to the Luminosity method.

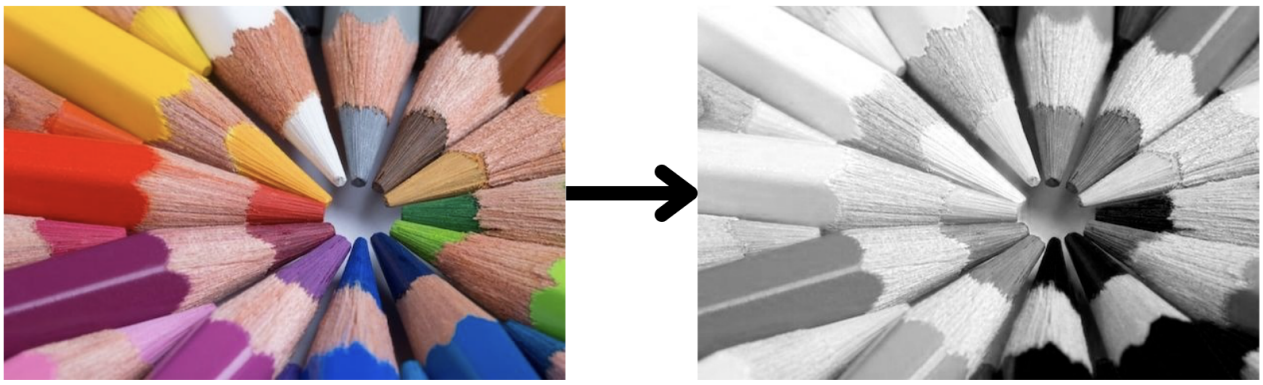


**4. Single Channel Extraction**

- **Method:** Use one of the color channels (Red, Green, or Blue) directly as the grayscale value.
- **Formula:**

$$Gray = R \text{ or } G \text{ or } B$$

- **Effect:** Extracting just one channel can highlight specific features of the image related to that color. For example, the red channel might emphasize warmer areas, while the blue channel could highlight cooler regions. The resulting grayscale image may have a very different appearance depending on which channel is chosen.

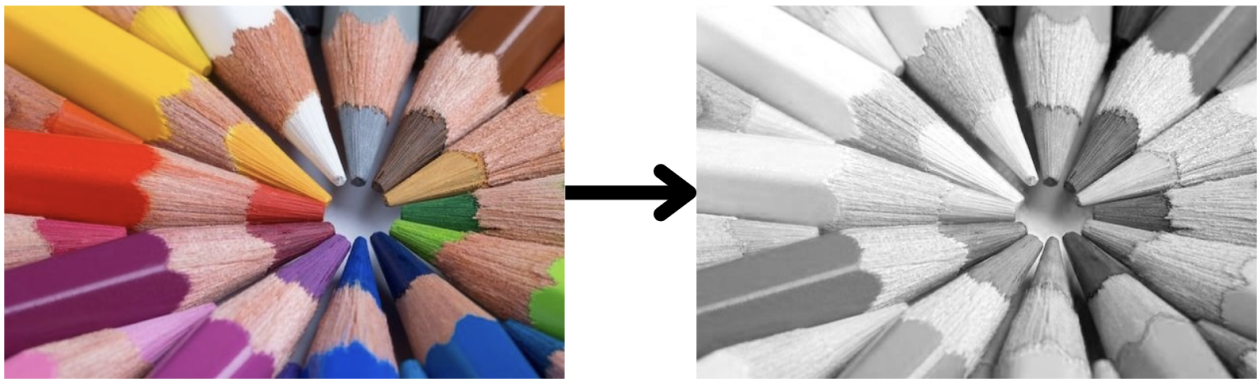


5. **Decomposition Method**

   - **Method:** Use either the maximum or minimum value of the RGB channels as the grayscale value.
   - **Formula:**

$$Gray_{max} = max(R, G, B) Gray_{min} = min(R, G, B)$$

   - **Effect:** Using the maximum value tends to produce a brighter image, while the minimum value results in a darker grayscale image. This method can be useful for specific effects, like emphasizing certain features.



6. **Principal Component Analysis (PCA)**

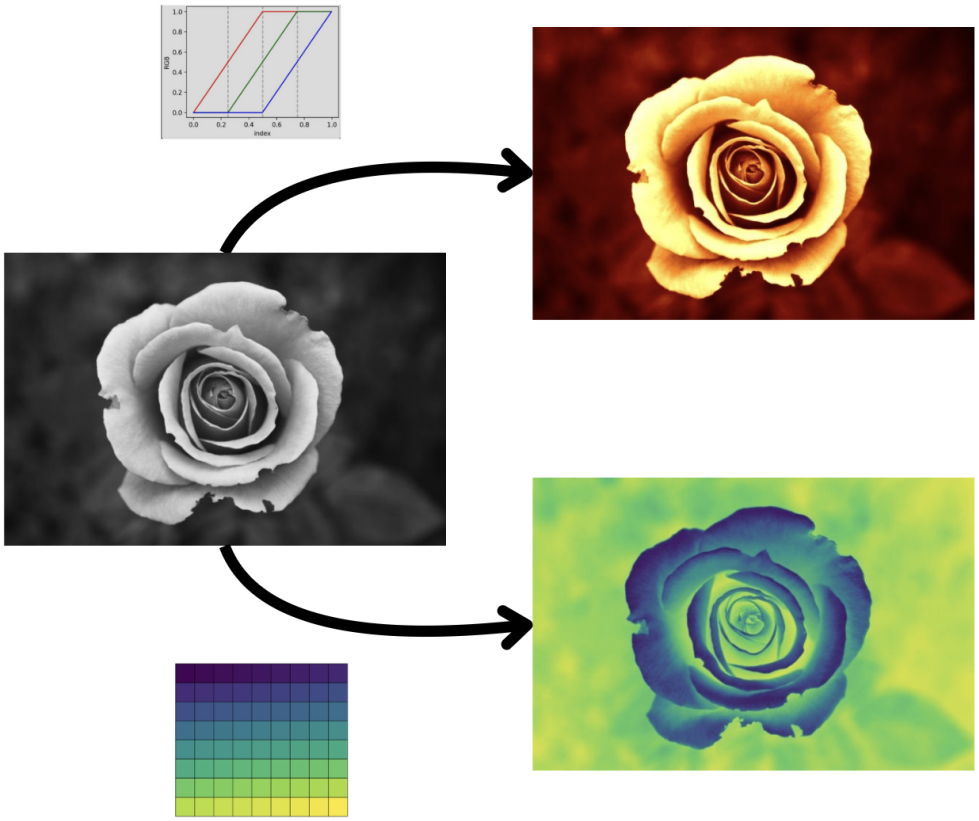
   - **Method:** Perform PCA on the RGB channels and use the first principal component as the grayscale image.
   - **Effect:** PCA-based grayscale conversion can capture the most significant variance in the image, potentially enhancing features or patterns that are not as prominent with other methods. However, it might also produce unexpected results depending on the image.

7. **Conversion Using Color Models**

   - **Method:** Convert the image to a different color model, such as HSV or LAB, and use the luminance or value component.
   - **Effect:** Depending on the color model used, this method can produce grayscale images that emphasize brightness (value) or lightness, often capturing features that are missed in the RGB space.

## Pseudo Color Mapping

1. In this I have deployed two different types of color maps.
2. One is based on linear functions for each of the channels and the other is based on interpolation of segment colors (colors provided for in-between points which are then used to interpolate to intermediate points) using VIRDIS color palette, as shown below:



## Replacing Green Screen

1. Iterated thorough all the pixels in the green screen image.
2. Identified all the pixels which are to be replaced based on certain threshold value for the green channel as well as minimum threshold for the difference between green and other channels.
3. For all the pixels which are identified as green (background) we replace them with the corresponding pixels from the background image.

**Note**

In the code the threshold for classification as background is $G \geq 200$ and $|R - G| > 50$ and $|B - G| > 50$

## Reading and Writing of Video

1. For reading a video in an array, we use cv2's VideoCapture function to read the video's data frame by frame and store frame array's in the final 4D list.
2. Similarly using cv2 we rewrite the 4D list into a ".mp4" file frame by frame.

## Transition Video using fade and slide effects

1. We take one start image and one end image.
2. In fade we gradually decrease the intensity of pixels of start image and increase the intensity of pixels of end image, frame after frame untill the start image is completely invisible and the end image is visible, using this formula:

$$F_t(x, y) = (1 - t)I_1(x, y) + tI_2(x, y)$$

$$t = n/N; \qquad n = 0, 1, ...N$$

3. For slide transition video, with each frame we are including 20 more columns from the final image to overwrite on the initial image. In this way it gives a slow flow of image swipe from left to right.

**Note**

I am keeping the frame rate of the video at 20 because otherwise the transition effect becomes so fast and is not visible properly.