The shape of the grayscale image:

Since the output of *image.shape* was *(183, 275, 3)*, this means our image is **color (RGB)** not grayscale yet, therefore we must first convert it.
The used image was loaded using *plt.imread('path')* so the RGB channels were saved as

Red=image[..., 0] Green=image[..., 1] Blue=image[..., 3]

and using *image[..., 3]* means that we select all pixels and only the first 3 channels (RGB).

When converting color to grayscale, not all color channels contribute equally to the perceived brightness because the **human eye is most sensitive to green light**, less to red, and least to blue, therefore:

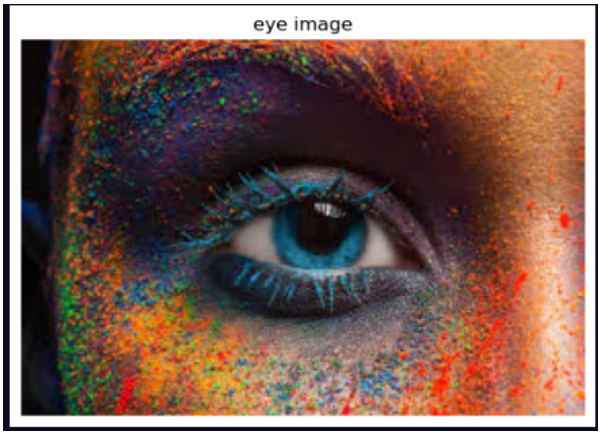
- **Red (R)** contributes about **29.89%**
- **Green (G)** contributes about **58.70%**
- **Blue (B)** contributes about **11.40%**

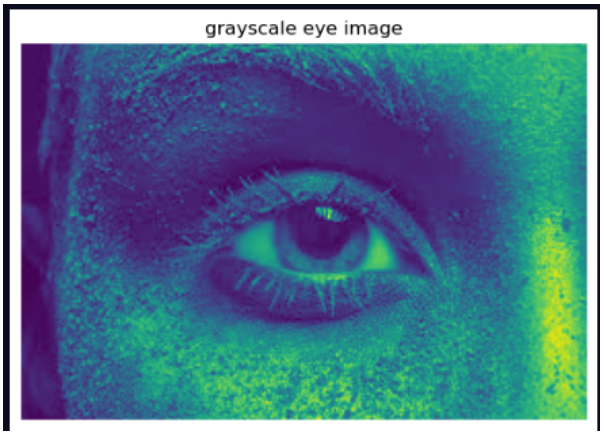
So the formula to convert from RGB to grayscale becomes:

gray = 0.2989*Red + 0.587*Green + 0.114*Blue

And the code becomes

gray = np.dot(image[..., :3], [0.2989, 0.587, 0.114]



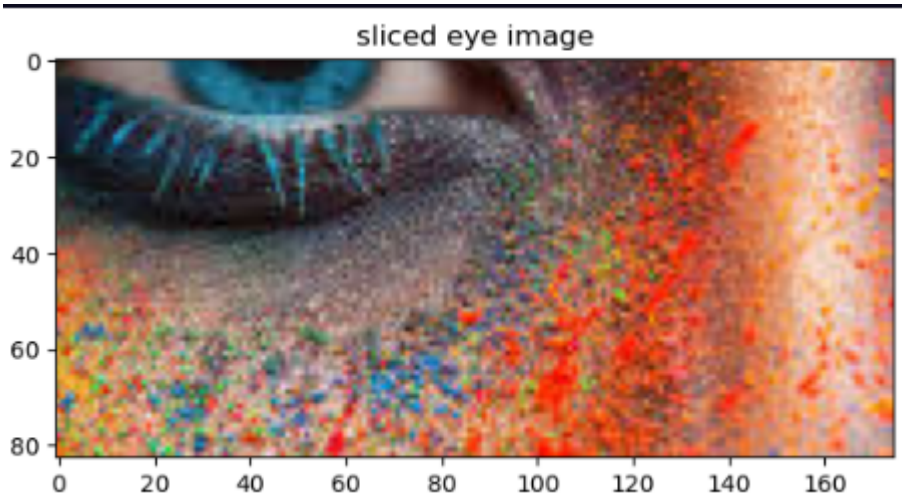


Slicing the image:

The code *plt.imshow(image[100:400, 100:400, :])* gives us a sub-image from the original one, the general form is:

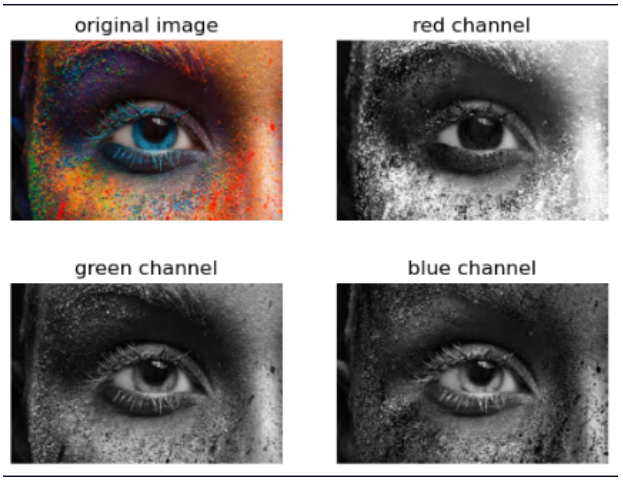
image[y_start:y_end, x_start:x_end, channels]

What we did is simply we selected pixels vertically then horizontally from index 100 to 399 with keeping all the channels (RGB), but since our image shape is (183, 275, 3) so the max row index is 182 and the max column index is 274 so slicing 100:400 will go out of bounds for both dimensions, this means effectively we get only image[100:183, 100:275, :] that's why we got a displayed patch with only (83*175) pixels and not (300*300).



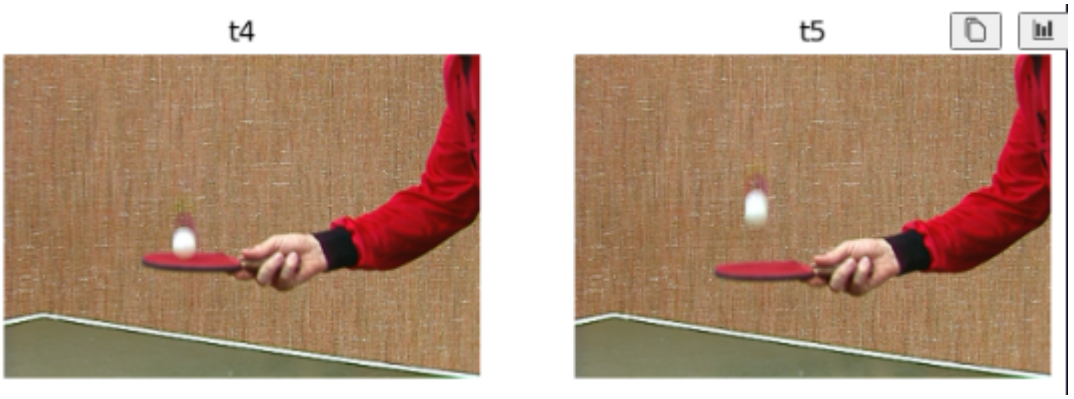
RGB channels:

After isolating each color layer from the original image array, each variable from red, green and blue has the intensity map for its own color channel, then we stacked back the separated channels which did give the original image.

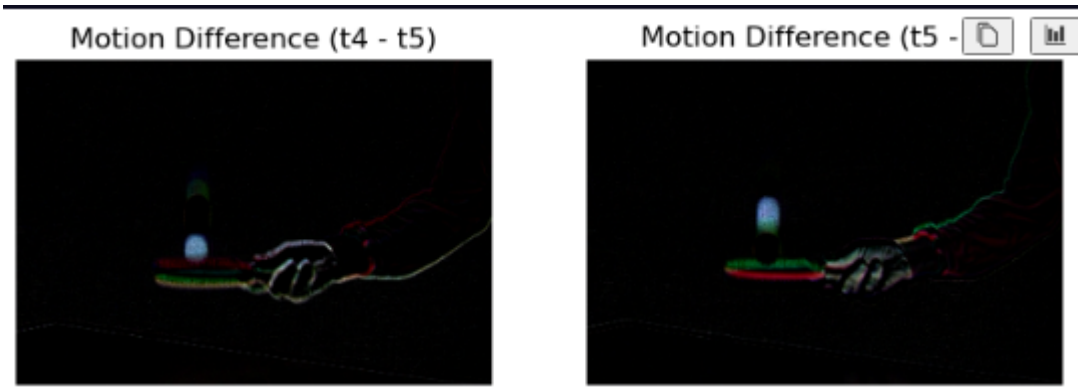


Motion Difference:

The goal is to detect motion or change between the two images t4 and t5 and by subtracting one image from the other, we highlight what has changed, moving objects or differences in brightness and so.



In order to compute the motion difference we subtract pixel by pixel as follows: difference54 = t5 - t4 and difference45 = t4 - t5 where each pixel represents how much the color intensity has changed



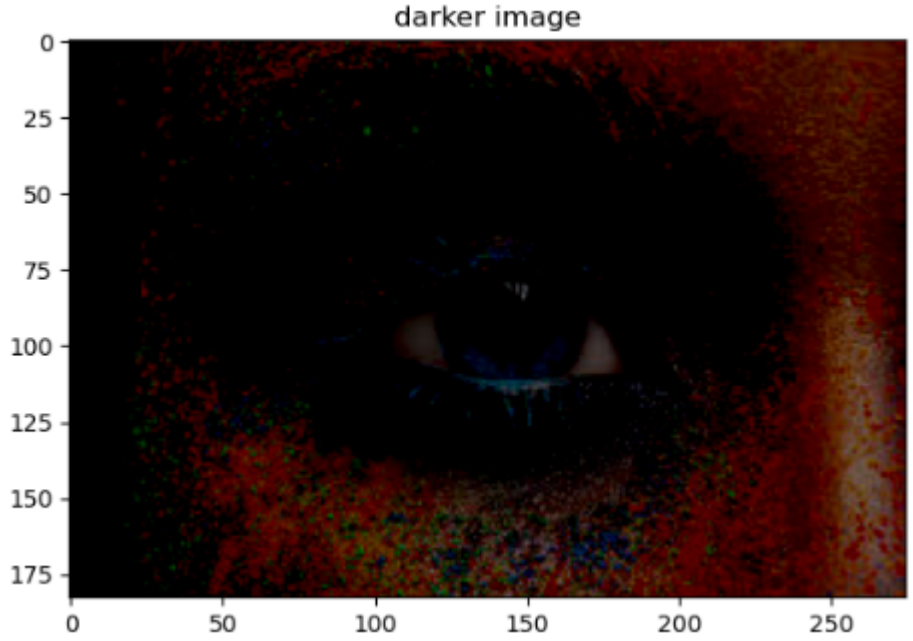
Point Processing Techniques:

1.Darker image:
We used the *skimage.color* library to convert the image from the RGB color space (which represents colors as combinations of red, green, and blue) to Lab color space because Lab separates luminance/lightness from color so we could reduce brightness without distorting colors, To do so, we converted RGB to Lab where:

- L channel refers to lightness (0-100)
- a, b channels refer to color components

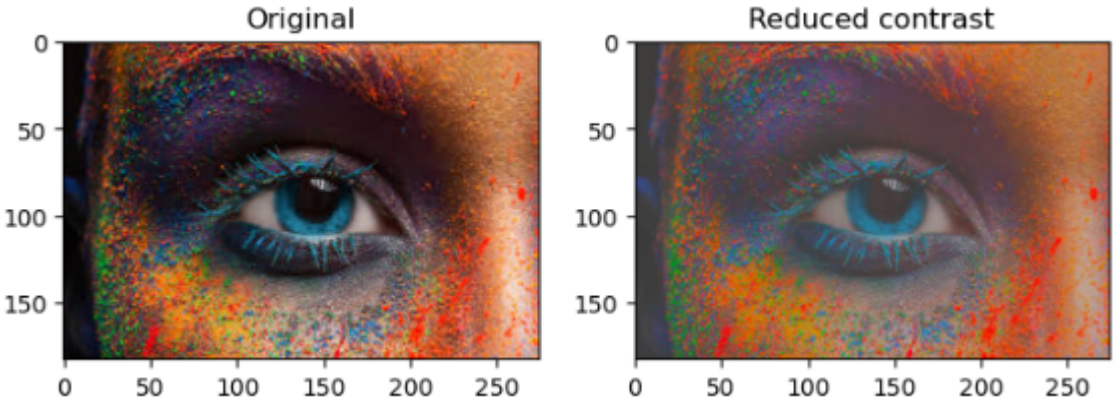
To decrease the brightness we did reduce the L component by 50 units which made the entire image darker to have a more darker image, then converted back to RGB.

```
darker = rgb2lab(image)
darker[...,0] = darker[...,0] - 50
darker = lab2rgb(darker)
```



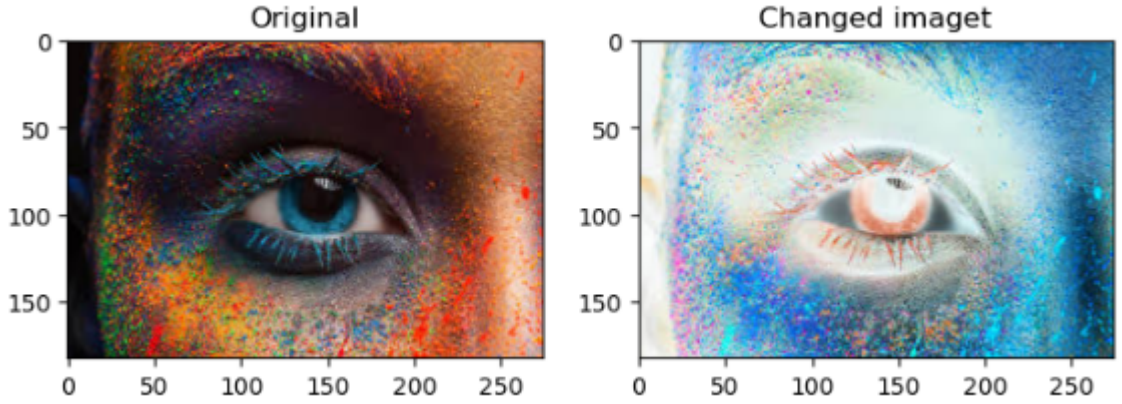
2.Reduce the difference between light and dark:

When reducing the difference between light and dark areas, the bright pixels become a bit darker and the dark ones also become a bit lighter. To do so, we applied a contrast compression on the L channel which pulls both extremes toward the mid-range of brightness, making illumination more uniform across the image.



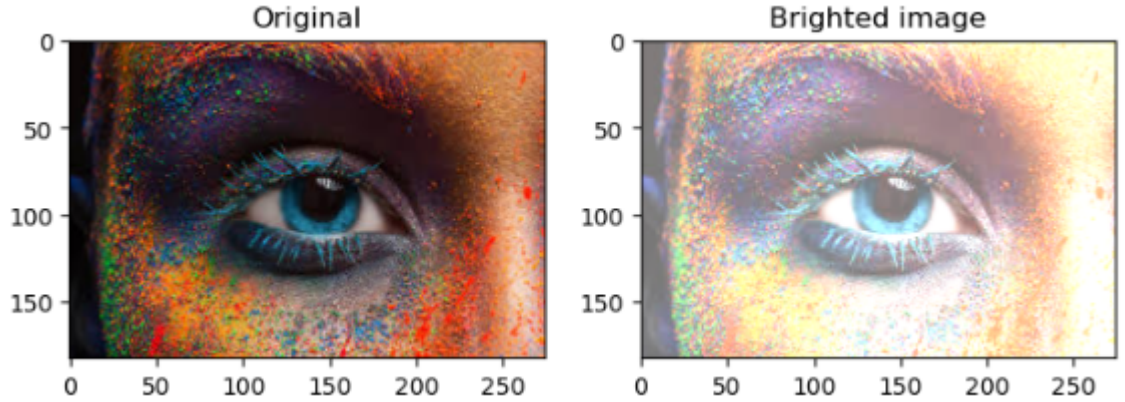
3.Change the pixel values from dark to light:

We need to invert the brightness of the image by making the dark areas become light and the light ones become dark which means the pixel that was very bright (close to white) becomes very dark (close to black) and the pixel that was dark (close to black) becomes bright.



4.Increase the brightness:

We used the image converted to Lab in order to manipulate the brightness without distorting colors then increasing the brightness channel by 50 which will move every pixel 50 units closer to white.



5.Increasing the difference between light and dark areas:

By multiplying the L channel by 2, the dark pixels become even darker and bright pixels become even brighter which will increase the difference between dark and light pixels.