Assignment #1

HARNEET SINGH (400110275)

January 21, 2022

Theory

1.1

Camera resolution shows the upper limit of number of pixels, i.e. picture elements, in an image taken by the camera. In the given smartphone specifications, there are three cameras on the back for specific purposes and on the front side (selfie camera), there is only one camera. Back cameras have resolutions of 50 MP, 12 MP and 40 MP respectively, and selfie camera has a resolution of 32 MP. NOTE: ($1\ Megapixels = 10^6\ pixels$).
A more advanced definition of resolution defines number of pixels (also known as photosites - capture amount of incoming light and produce data for the image) on the surface of the digital camera's sensor (Bayer - CFA). Although higher resolution yields better image quality (finer details), but it does not always lead to a better performing camera which depends on other factors as well.
Another important point to note is that the image resolution and video resolution on the same device differ from one another.

1.2

Pixel size is the dimension of a pixel on a camera sensor and it is used to retrieve light information when taking an image. Obviously, larger the pixel size, the more light it can receive which provides better image quality, especially in low light conditions. As the resolution (number of pixels) increases on a sensor, the pixel size decreases which can impact the image quality.
In the given specifications, the main camera's pixel size is 2.44 µm (microns) while the selfie camera's pixel size is 2.44 µm. Clearly, the main camera will outperform the selfie camera in a dark settings, because the pixel size of main camera is larger in comparison to selfie camera's pixel size.
As the pixel size diminishes, there is a higher chance of capturing digital noise. A binning technique is deployed which utilizes multiple pixels to form a single pixel 'super-pixel' i.e. light information from multiple sensors is combined to form a single pixel.

1.3

PDAF stands for Phase Detection Auto Focus which is a technique used to auto focus the camera lens on an object. In phone cameras, the concept of PDAF works with the help of paired photo-diodes which are physically masked to receive light from only one direction. The photo-diode pair is placed close to one another so that they receive same amount of light and each photo-diode allows for light from only one side to pass through i.e. first photo-diode will allow light from left side (and block the light from right side) and the second photo-diode will allow the light from right side (and block the light from left side). This way two images are created and compared to analyse if the object is into focus. If the two images created by the pair of photo-diodes are not similar, the phase difference is calculated to figure out how much lens movement will bring the image into focus.
Generally, hybrid method is used to auto focus the camera quickly and accurately i.e. both CDAF (Contrast Detection Auto Focus) and PDAF are employed. However, PDAF is much faster than CDAF (a more conventional method used in DSLRs).

1.4

Shutter speed represents the amount of time for which the sensor is allowed to receive light. Usually, it is represented in seconds (e.g. shutter speed of 1/30s). In mobile phones, the shutter mechanism is achieved by turning the sensors on and off for a certain amount of time and CMOS sensors are used which turn the sensors ON in a sequence of top-left to bottom-right. Unlike mechanical shutters, mobile camera's shutter (rolling shutter) does not open (turn ON) concurrently for the entire lens, i.e. sensor captures light one row at a time.

As we increase the shutter speed, we lose the amount of light that is incident on the sensor (more light produces brighter image). That is why with higher shutter speed, the image will be darker. On the other hand, with faster shutter speed, the electronic shutter can mimic the mechanical shutter (mechanical shutter opens at once for a certain amount of time and exposes the entire lens to capture light concurrently). Another advantage is that with higher shutter speed, the objects in motion can be captured without any motion blur, and the image will be sharp.

1.5

OIS stands for Optical Image Stabilisation. OIS is used to counteract slight movements or jitters caused while taking a picture or a video. It is achieved with the help of gyroscope by adjusting the position of the lens or the sensor in case the camera is shaking. Having OIS has been extremely helpful in taking good quality images because the physical hardware compensates for minor hand movements and thereby, offers crisp and blur-free images.
Without the OIS, if we use lower shutter speed which means that the shutter stays open for a longer period of time, then any movements induced by our hand (or a tripod stand) will be registered on the sensor, thus producing a blurry image. However, if we use OIS, then the minor jitters will be counteracted by the OIS system which will allow a better quality image because lower shutter speed (more light) and less vibrations (steady sensor) means a crisp image.

1.6

ISO stands for International Organization for Standardization. ISO sensitivity describes the amount of light needed to expose the lens for good image. ISO sensitivity is represented with integer numbers, usually ranging from 50 to values in thousands. As the ISO number increases, less light is required. This is why low ISO values are used in bright spots and large ISO values are used in dark spots. As the ISO value is increased, it requires less light therefore shutter speed can be reduced. Care should be taken when manually changing the ISO value because a large ISO value against a bright scene may produce a grainy image. This is a result of generating noise by allowing too much light on the sensor.

2

Gamma correction is used to display an output image at an intended luminance. Gamma correction is a two stage process which involves gamma encoding and gamma decoding to store and view the images at the desired output quality (brightness). Because, human eye is more perceptible to changes in the darker tones in comparison to brighter tones, more bits are needed to represent darker tones than the brighter tones. This process of encoding and decoding (correcting) allows the image to be produced as the original color setting (or even, at better brightness level) and it is done with the help of following formula:

$$V_{out} = A * V_{in}^{\gamma}$$

where, A is usually equal to 1.

Clearly, the final result of gamma correction can be easily linearized by applying the inverse of gamma encoded value to the stored image data. This will ensure that the reproduced image is close to the linear line i.e. line with γ equal to 1 which represents the original scene. Gamma correction is completed on the image before inputting it to the monitor.

Plot of output value vs. input value is shown below for $\gamma = \{0.25, 0.5, 1, 1.5,2\}$ (Assumption: A = 1):
V_{in} is in the range of 0 to 255 and V_{out} is scaled after computing the product of A and V_{in}^{γ} such that range of V_{out} is between 0 and 255 as well. Each curve is scaled independently so that it would fit in the same graph.

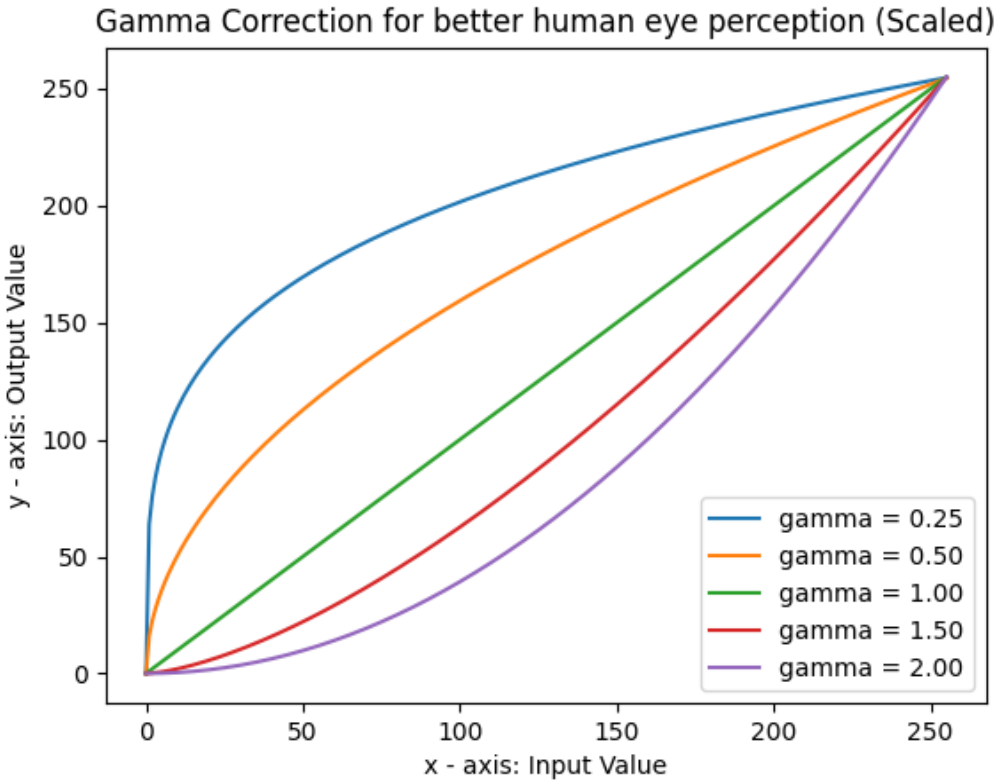

Figure 1: Gamma Correction with $\gamma = \{0.25, 0.5, 1, 1.5,2\}$

3

As we already know, a color can be described by its two fundamental properties: luminosity and chromaticity.

XYZ color space is developed by CIE (Commission Internationale d'Eclairage) to represent all the colors perceived by humans. XYZ values are calculated using the spectral intensity for light of all wavelength and in a way, it represents the tristimulus values. It is the primary color space i.e. other color spaces, such as RGB, are formed using XYZ color space. Technically speaking, the gamut of other color spaces are usually contained within gamut of XYZ color space. It describes the color by adding the primary colors. It should be noted that Y in XYZ closely represents the luminous (brightness of color).

xyY color space describes a color with the notion of luminous and chromaticity. x and y tell us about the chromaticity of the color and Y tells us about the luminance. Following formulae can be used to figure out xyY values:

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z}$$

Note that x, y and z values are normalized and their sum equals to 1. So, we can find the value of any two components if two of them are known, e.g., by using $y = 1 - x - z$. However, z value is ignored because x and y tell us about the chromaticity of the color. As stated earlier, Y represents the luminosity value and is retrieved from the original XYZ color space.

Chromaticity diagram shows all the colors percieved by humans on an x-y plane. Chromaticity diagram is in the shape of a horseshoe and the outer circumference represents all of the pure monochromatic color values. The line that connects the two endpoints is called purple line and usually an 'E' letter is shown to mark the white point. From within this chromaticity diagram, we can obtain RGB colors as well which is mapped as a triangle.
From the gamut (scope of chromaticity diagram), we can see that all the values (x, y, z) are positives. This diagram shows all of the chromaticities seen by human eye. Another interesting point is that if we create a straight line using two points on the gamut, then we can create all the colors on the line by mixing the two endpoint colors. Similarly,

this concept is applicable with three points forming a triangle as well i.e. three vertices of a triangle can form all of the colors within it. Also, note that no triangle can be formed that can cover the entire gamut of this chromaticity diagram.
Another fact is that colors inside the diagram can be formed in different ways, except the monochromatic colors on the outer rim.

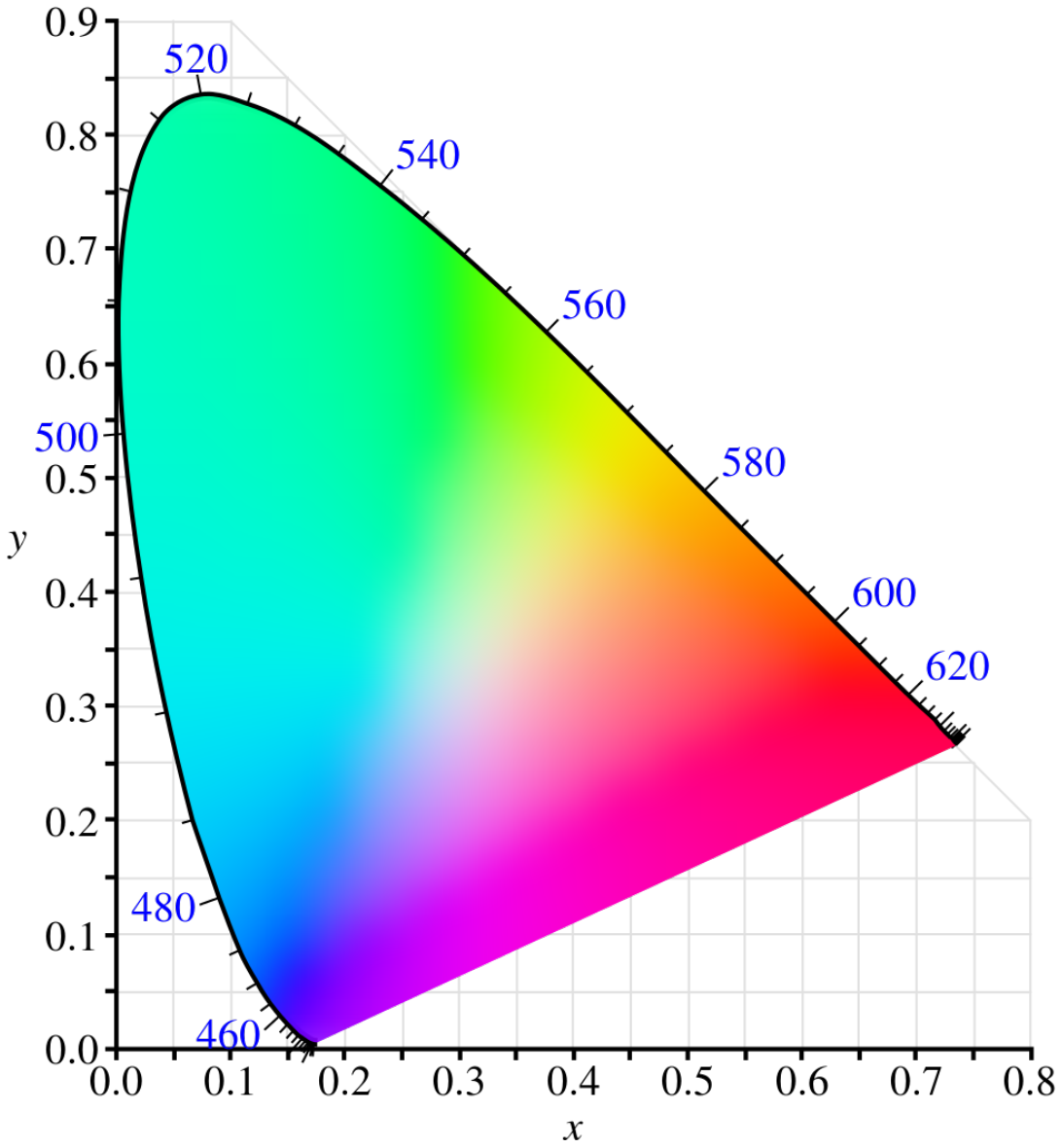

Figure 2: Chromaticity Diagram [2]

All of the numbers (in blue color) are in nano-meter (different wavelengths of colors in light)

4

Bilinear interpolation is a re-sampling technique that allows to obtain a pixel value from its nearest four pixel values (known). It is similar to linear interpolation, except it is carried out on both x and y directions and the pixel value is weighted-average and is based on the distance between the neighboring four pixels. Bilinear interpolation gives us an opportunity to obtain pixel values for the locations where pixel values are unknown and this interpolation is useful when the image is zoomed, rotated and geometrically corrected.
This is beneficial to us when upscaling the image, because after bilinear interpolation, we have more pixel values than the original image and the image would not seem grainy when zoomed in. Intuitively, interpolation method creates a bigger grid of pixels than the original grid of pixels. In a nutshell, bilinear interpolation provides a pixel value between known four pixels, as shown in the example below.

Formula used for bilinear interpolation is as follows:

$$Q(x, y) = ax + by + cxy + d$$

As one can imagine, with four known pixel values, we can get four equations using the above-stated formula to retrieve coefficient values (a, b, c, d). Given: $x_1 = 10, x = 20, x_2 = 50, y_1 = 10, y = 30, y_2 = 40, Q_{11} = 10, Q_{12} = 100, Q_{21} = 60, Q_{22} = 70$

1. Using Polynominal Fit (approximation), we get:

$$Q_{11} : 10 = 10a + 10b + 100c + d$$

$$Q_{12} : 100 = 10a + 40b + 400c + d$$

$$Q_{21} : 60 = 50a + 10b + 500c + d$$

$$Q_{22} : 70 = 50a + 40b + 2000c + d$$

   Solving above equations give us: $a = \frac{23}{12}, b = \frac{11}{3}, c = \frac{-1}{15}, d = \frac{-235}{6}$

   So, in this case, the bilinear equation becomes:

$$Q(x, y) = \frac{23}{12}x + \frac{11}{3}y - \frac{1}{15}xy - \frac{235}{6}$$

   Therefore,

$$Q_P = Q_{(20,30)} = \frac{23}{12} * 20 + \frac{11}{3} * 30 - \frac{1}{15} * 20 * 30 - \frac{235}{6} = \left\lfloor \frac{415}{6} \right\rfloor = 69$$

   So, the pixel value at point P is 69 which seems true to the fact that it is weight-average of its neighboring pixels.

2. Using Repeated Linear Interpolation, we get:

$$Q_{R_1} = \frac{x_2 - x}{x_2 - x_1}.Q_{11} + \frac{x - x_1}{x_2 - x_1}.Q_{21} = \frac{30}{40}.10 + \frac{10}{40}.60 = 22.5$$

$$Q_{R_2} = \frac{x_2 - x}{x_2 - x_1}.Q_{12} + \frac{x - x_1}{x_2 - x_1}.Q_{22} = \frac{30}{40}.100 + \frac{10}{40}.70 = 92.5$$

   Now that we have interpolated pixel values along x-axis, we will do the same along y-axis while using Q_{R_1} and Q_{R_2} values:

$$Q_P = \frac{y_2 - y}{y_2 - y_1}.Q_{R_1} + \frac{y - y_1}{y_2 - y_1}.Q_{R_2} = \frac{20}{30}.(92.5) + \frac{10}{30}.(22.5) = \lfloor 69.17 \rfloor = 69$$

   Both methods give us the same result.

Implementation

1 - Hello CV

- Angle 0° -

```python
import cv2

# reading the image as is and storing it in a variable
img = cv2.imread('img1.png', cv2.IMREAD_UNCHANGED)
# print(img)

#show image in a window named "img1"
cv2.imshow("img1", img)

#press 's' if you want to save image, else press any other key
#later, destroy all the windows to free up the resources
if cv2.waitKey(0) == ord('s'):
    cv2.imwrite("img1_0.png", img)
    cv2.destroyAllWindows()
else:
    cv2.destroyAllWindows()
```

Figure 3: Angle 0° Code

Figure 4: Angle 0° Result (Scaled Down)

- Angle 90° -

```python
import cv2

# reading the image as is and storing it in a variable
img = cv2.imread('img1.png', cv2.IMREAD_UNCHANGED)
# print(img)

#rotating image counterclockwise by 90 degrees
img = cv2.rotate(img, cv2.cv2.ROTATE_90_COUNTERCLOCKWISE)
cv2.imshow("img1", img)

#if image processed correctly, it will be saved by pressing 's',
#otherwise quit the window and free up the resources
if cv2.waitKey(0) == ord('s'):
    cv2.imwrite("img1_90.png", img)
    cv2.destroyAllWindows()
else:
    cv2.destroyAllWindows()
```

Figure 5: Angle 90° Code

Figure 6: Angle 90° Result (Scaled Down)

- Angle 180° -

```python
import cv2

# reading the image as is and storing it in a variable
img = cv2.imread('img1.png', cv2.IMREAD_UNCHANGED)
# print(img)

#rotating image by 180 degrees
img = cv2.rotate(img, cv2.cv2.ROTATE_180)
cv2.imshow("img1", img)

#if image processed correctly, it will be saved by pressing 's',
#otherwise quit the window
if cv2.waitKey(0) == ord('s'):
    cv2.imwrite("img1_180.png", img)
    cv2.destroyAllWindows()
else:
    cv2.destroyAllWindows()
```

Figure 7: Angle 180° Code

Figure 8: Angle 180° Result (Scaled Down)

- Angle 270° -

```python
import cv2

# reading the image as is and storing it in a variable
img = cv2.imread('img1.png', cv2.IMREAD_UNCHANGED)
# print(img)

#rotating image by 270 degrees
img = cv2.rotate(img, cv2.cv2.ROTATE_90_CLOCKWISE)
cv2.imshow("img1", img)

#if image processed correctly, it will be saved by pressing 's',
#otherwise quit the window
if cv2.waitKey(0) == ord('s'):
    cv2.imwrite("img1_270.png", img)
    cv2.destroyAllWindows()
else:
    cv2.destroyAllWindows()
```

Figure 9: Angle 270° Code

Figure 10: Angle 270° Result (Scaled Down)

2 - Gamma Correction

```python
import cv2
import numpy as np

# reading the image as is and storing it in a variable
img = cv2.imread('img1.png', cv2.IMREAD_COLOR)
# print(img.shape)

gamma = 0.60

#Applying the gamma correction formula using numpy array
#Scaling down to apply the formula and then, scaling up
#to range between 0 and 255 (integer)
gammaCorrectedArray = np.array( ((img/255)**gamma) * 255, dtype = 'uint8' )
# print(gammaCorrectedArray.shape)
# print(gammaCorrectedArray)

cv2.imshow("img2_gamma", gammaCorrectedArray)

#if image processed correctly, it will be saved by pressing 's',
#otherwise quit the window
if cv2.waitKey(0) == ord('s'):
    cv2.imwrite("img2_gamma.png", gammaCorrectedArray)
    cv2.destroyAllWindows()
else:
    cv2.destroyAllWindows()
```

Figure 11: Gamma Correction Code

Figure 12: Gamma Correction Result (Scaled Down)

2 - Skin Detection

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# reading the image as is and storing it in a variable
img = cv2.imread('selfie1.jpg', cv2.IMREAD_COLOR)
# print(img.shape)

#converting the color space to HSV from BGR
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

#calculating and plotting the histogram to discover the hue range
hist_hue = cv2.calcHist([hsv_img], [0], None, [180], [0, 180])
# print(hist_hue)
plt.plot(hist_hue, color='r', label = 'hue')
plt.xlim([0, 180])
plt.legend()
plt.show()

# define range in HSV
lower_range = np.array([0,30,30])
upper_range = np.array([20,255,255])
# Threshold the HSV image to get only skin color
mask = cv2.inRange(hsv_img, lower_range, upper_range)
# Bitwise-AND mask and original image
result = cv2.bitwise_and(img, img, mask = mask)
cv2.imshow('HUE', result)

# if image processed correctly, it will be saved by pressing 's',
#otherwise quit the window
if cv2.waitKey(0) == ord('s'):
    cv2.imwrite("impl3.png", result)
    cv2.destroyAllWindows()
else:
    cv2.destroyAllWindows()
```

Figure 13: Skin Detection Code

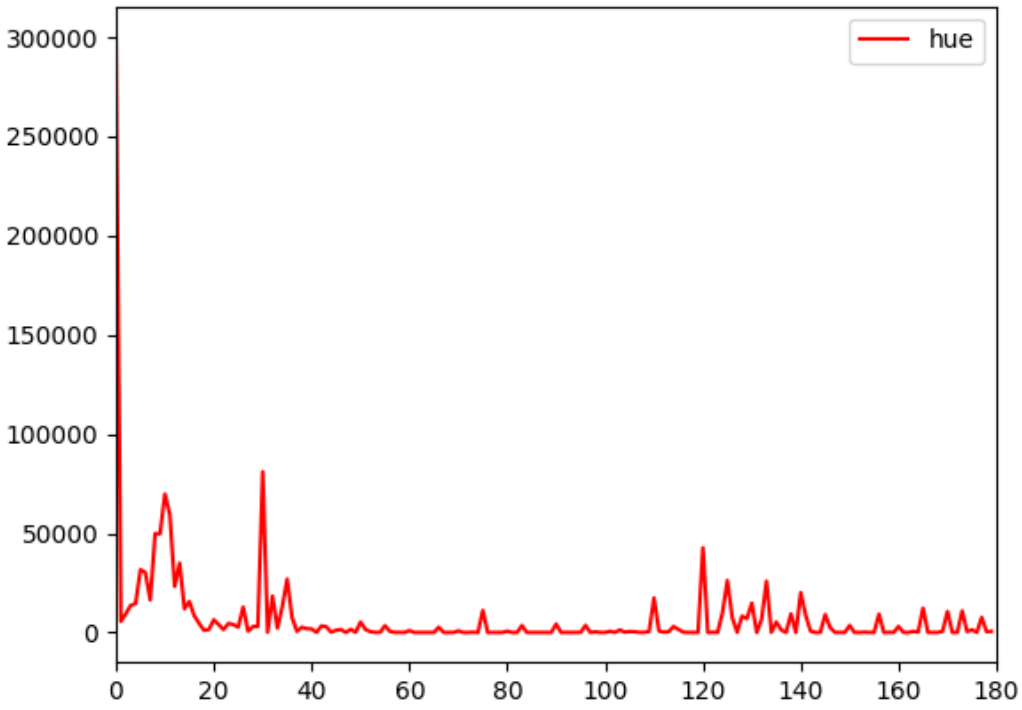

Figure 14: Skin Detection Histogram (Hue)

As one can see, hue values are quite large in the interval of 0 to 20 i.e., in the code, this range is chosen to detect the skin. Therefore, threshold $\in$ [0, 20].

Figure 15: Selfie for Skin Detection

Figure 16: Skin Detection Result

References

[1] Bilinear interpolation. https://en.wikipedia.org/wiki/Bilinear_interpolation.

[2] Cie1931xy blank.svg. https://en.wikipedia.org/wiki/CIE_1931_color_space.

[3] Gamma correction. https://learnopengl.com/Advanced-Lighting/Gamma-Correction.

[4] Introduction to light, color and color space. https://www.scratchapixel.com/lessons/digital-imaging/colors/color-space.

[5] Cie 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space, January 2022.

[6] Gamma correction. https://en.wikipedia.org/wiki/Gamma_correction, January 2022.

[7] Understanding iso sensitivity. https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-iso-sensitivity.html, January 2022.

[8] R.C. Gonzalez and Richard E. Wods. *Digital Image Processing*. Pearson, 2017.

[9] Gav L. Everything you need to know about shutter speed on mobile cameras. https://thesmartphonephotographer.com/mobile-camera-shutter-speed/.

[10] Gav L. How does a smartphone camera work? a detailed walk through. https://thesmartphonephotographer.com/how-phone-camera-works/.

[11] Gav L. Phone camera megapixels explained: The real truth. https://thesmartphonephotographer.com/truth-about-phone-megapixels/.

[12] Gav L. The smartphone photographer. https://thesmartphonephotographer.com/phone-camera-specs-explained/, May 2020.

[13] NASIM MANSUROV. What is iso? the complete guide for beginners. https://photographylife.com/what-is-iso-in-photography, August 2019.

[14] Udi Tirosh. Everything you wanted to know about rolling shutter. https://www.diyphotography.net/everything-you-wanted-to-know-about-rolling-shutter/, September 2012.

[15] Robert Triggs. What is pdaf and how does it work? phase detection autofocus explained. https://www.androidauthority.com/how-pdaf-works-1102272/, May 2021.

[16] Robert Triggs and Ryan-Thomas Shaw. What is image stabilization? ois, eis, and his explained. https://www.androidauthority.com/image-stabilization-1087083/, November 2021.