4TN4 Assignment 1

Adam Bujak

January 21, 2022

1 Theory

1.1 Phone Camera Parameters

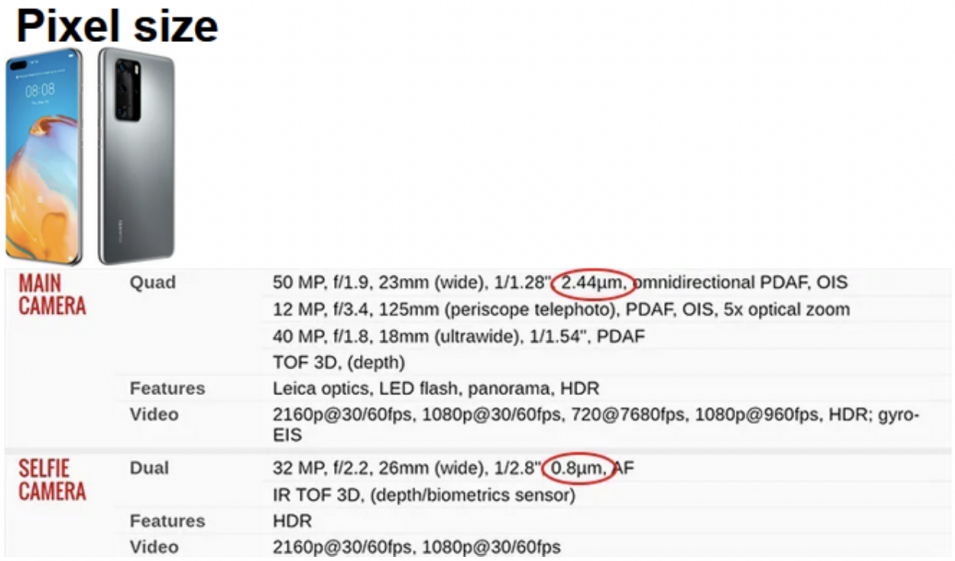

Figure 1: Phone Specifications

1.1.1 Resolution

What does the camera resolution show?

The main camera (wide) is a 50MP camera - indicating that there are 50 million pixels. This is simply a measure of how many RGB measurements are recorded for each photograph.

1.1.2 Sensor Size

As shown the size of the main camera is 2.44µm while the selfie camera sensor size is 0.8µm. First define the camera sensor size and the state what is the advantage of having larger sensor size?

The 2.44µm measurement refers to the size of each individual pixel's size. The advantage of having a larger sensor is that you have more light sensing area per each pixel,

therefore, you can capture more light, and more importantly, reduce the noise in the light measurement on a per pixel basis.

1.1.3 PDAF

What does PDAF stand for? Explain shortly how it works?

PDAF stands for *Phase Detection Autofocus* which works by splitting the image into two phases using prisms, and then adjusting the lens until the two phases are in sync - this is when the image is in focus.

1.1.4 Shutter Speed

What is shutter speed? What do we lose and gain when we take a photo using high shutter speed?

Shutter speed is defined by how quickly the shutter opens and closes. Perhaps a more intuitive and practical term for this is *exposure time*, especially in modern times where more and more cameras do not have a physical shutter. Exposure time is the time for which sensor readings are made during an exposure (taking a photograph). Longer exposure times increase the time of exposure and increase the amount of light available for measurement. However, in well lit environments too long of an exposure will result in burn in (the imaging equivalent of audio clipping), since the maximum sensor reading will be reached. Too long of an exposure will also result in motion blur if objects in the scene are moving. Too short of an exposure time (high shutter speed) will result in not as much light being available for measurement and may result in an under-exposed image, however, a higher shutter speed will result in less motion blur and therefore a sharper image if the environment is in motion.

1.1.5 OIS

What does OIS stand for? Using OIS, we can use a lower shutter speed. How could this improve the quality of the taken image?

OIS stands for *Optical Image Stabilization*. Camera systems that use OIS physically stabilize either the sensor or the lens which reduces the motion blur effect due to the camera system shaking while being operated. By reducing the movement of the image on the sensor, we can get a clearer, sharper image at slower shutter speeds.

1.1.6 ISO

What is ISO sensitivity in cameras?

ISO sensitivity essentially determines the gain for amplifying the light readings. A high ISO will result in a brighter image but higher noise levels. A low ISO will result in a dimmer image (thus requiring longer exposure times) but a much clearer image with less noise.

1.2 Gamma Correction

Why gamma correction is useful? present the formula of gamma correction and plot the output value vs. input value (assume the images are between 0 and 255) for $\gamma = \{0.25, 0.5, 1, 1.5, 2\}$

Camera sensors are linear - when two units of light hit the sensor, the pixel will be twice as bright as when one unit of light hits the sensor. This is not always the case in humans. Especially in darker scenarios where humans are much more sensitive to changes in brightness. Gamma correction corrects for this with the following formula, where P is a given pixel value.

$$P' = (\frac{P}{255})^{\frac{1}{\gamma}} \cdot 255$$

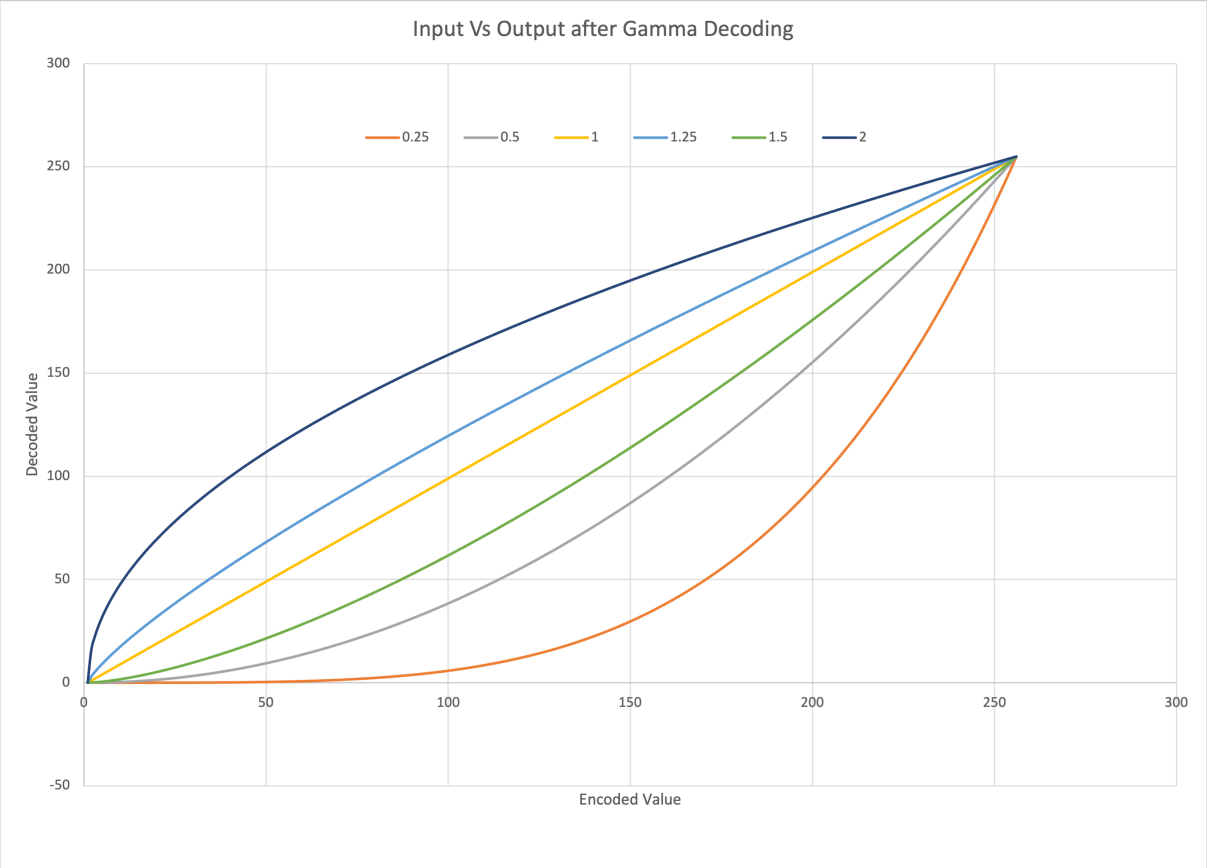

Figure 2: Gamma Correction with varied gamma values

1.3 Colour Spaces

What is XYZ and XYy color spaces? Explain what is chromaticity diagram and plot it.

1.3.1 XYZ

XYZ colour space is the representation of all possible colours the average human can see (experimentally observed by varying test colour wavelength) where X, Y, and Z correspond to light functions that are purely theoretical. When researchers attempted to recreate colours of certain wavelengths using real red, green, and blue colours, they found that they had to "subtract" certain colours to recreate the test colour. To avoid the "subtraction"

of light, which is impossible, they developed the XYZ basis which avoids this issue, by allowing the recreation of any colour through pure addition.

1.3.2 xyY

xyY is the normalized version of XYZ where the x and y are defined by the following equations, and Y remains the same to represent only luminance. Therefore, x and y completely represent the chromaticity of a colour, and Y represents the luminance.

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

1.3.3 What is a chromaticity diagram?

The chromaticity diagram of the CIE 1931 (XYZ) colour space is a visual representation of all of the colours visible to the average human. One can observe all of the possible colours that can be achieved by mixing two colours by connecting the two colours with a straight line. This further allows one to easily see how any colour can be created in an infinite number of ways, because as long as the path between the two colours used passes through the desired colour, it is a possible combination to recreate the desired colour.

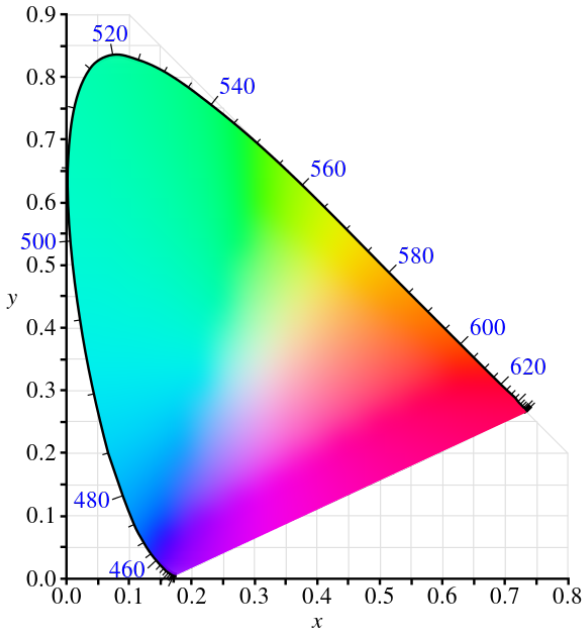

Figure 3: Chromaticity Diagram of the CIE 1931 (XYZ) colour space

1.4 Interpolation

What is bilinear interpolation? How can it be used to upscale (upsample) an image?

Linear interpolation is the process of estimating a value between two samples by drawing a straight line between the two samples and taking the value of the line at the

time of interest. For instance if we know a drone was 5m above the ground at $t = 0s$ and 10m above the ground at $t = 2s$, we can estimate that the drone was 7.5m above the ground at $t = 1s$.

Bilinear interpolation is the same idea but elevated into two dimensions. One way we can perform bilinear interpolation is by performing linear interpolation in one dimension first, and then using the values of the first interpolation, use linear interpolation again in the second dimension.

Bilinear interpolation can be used to upsample an image by adding more pixels by performing bilinear interpolation on the surrounding pixels. This would provide an estimate for the surrounding pixels, and will increase the resolution of the image. However, these extra pixels are only estimates, and are no replacement for a higher resolution original image.

Use bilinear interpolation to derive the green pixel value. (Q11 = 10,Q12 = 100,Q21 = 60,Q22 = 70)

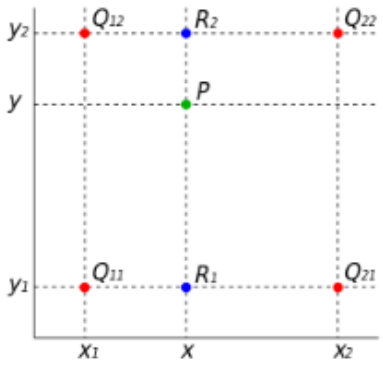

Figure 4: Bilinear Example Problem

$\boldsymbol{x_1 = 10, x = 20, x_2 = 50, y_1 = 10, y = 30, y_2 = 40}$

To find P using bilinear interpolation, we can find R_1 and R_2, using linear interpolation, and then linearly interpolate between R_1 and R_2.

$$R_1 = \frac{Q_{21} - Q_{11}}{x_2 - x_1} \cdot (x - x_1)$$

$$R_2 = \frac{Q_{22} - Q_{12}}{x_2 - x_1} \cdot (x - x_1)$$

$$P = \frac{R_2 - R_1}{y_2 - y_1} \cdot (y - y_1)$$

By plugging in, we find $R_1 = 12.5$, $R_2 = -7.5$, and $P = -13.\overline{3}$.

2 Implementation

2.1 Hello OpenCV

2.1.1 Code Screenshot

```
import numpy as np
import cv2
import time

def rotate(img, degree):
    # if degree is not 0, 90, 180, 270 throw exception
    if degree % 90 != 0:
        raise ValueError('invalid angle')

    if degree == 0:
        return img

    # rotation code
    rot = int(degree / 90) - 1

    newimg = []
    # rotate image
    return cv2.rotate(img, rot)

# import image
img = cv2.imread('img1.png')

# get rotation angle
rotangle = int(input("enter rotation angle: "))
img = rotate(img, rotangle)

# display image
cv2.imshow('image', img)
# wait for 3 seconds
cv2.waitKey(3000)
# close image
cv2.destroyAllWindows()
# save image
cv2.imwrite('rot{}.png'.format(rotangle), img)
```

Figure 5: Hello OpenCV Python Code

2.1.2 Results

Figure 6: 0 Rotation

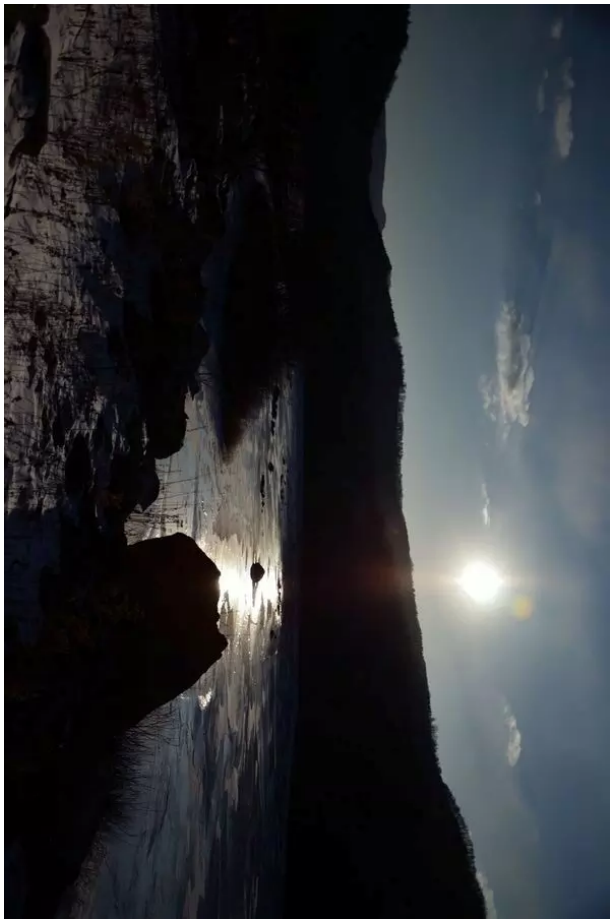

Figure 7: 90 Rotation

Figure 8: 180 Rotation

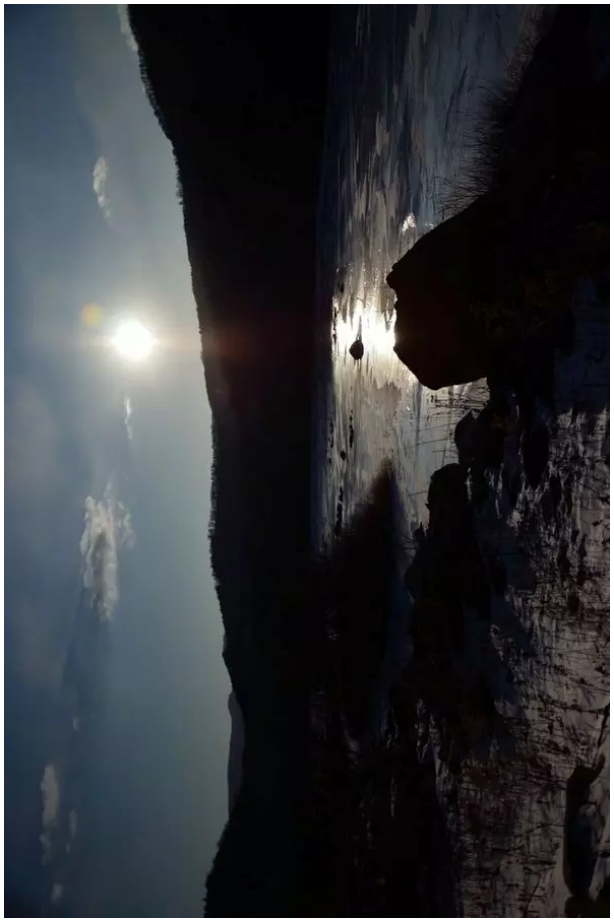

Figure 9: 270 Rotation

2.2 Gamma Correction

2.2.1 Code Screenshot

```
import numpy as np
import cv2
import time

def gamma_correct(img, gamma):
    # loop through each pixel and apply gamma correction
    for i in range(len(img)):
        for j in range(len(img[i])):
            # loop through each BGR channel
            for k in range(len(img[i][j])):
                img[i][j][k] = (img[i][j][k]/255)**(1/gamma)*255
    return img


# import image
img = cv2.imread('img1.png')
# gamma correct
img = gamma_correct(img, 2)

# display image
cv2.imshow('image', img)
# wait for 3 seconds
cv2.waitKey(3000)
# close image
cv2.destroyAllWindows()
#save img
cv2.imwrite('gamma_correction.png', img)
```

Figure 10: Gamma Correction Python Code

2.2.2 Results

Figure 11: Gamma Corrected image with $\gamma = 2$

2.3 Skin Detection

2.3.1 Code Screenshot

```python
import numpy as np
import cv2
import time

def detect_face(img):
    # convert to hsv
    img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # hue is represented as angle/2 in OpenCV
    # skin is in between 0 and 50 => 0 - 25 in OpenCV
    lower_hue = 1  # boost to 1 to remove some of red jacket in background
    upper_hue = 10 # lower to 10 to remove some of background

    # loop through every pixel
    for i in range(len(img)):
        for j in range(len(img[i])):
            if img[i][j][0] <= lower_hue or img[i][j][0] >= upper_hue:
                # remove pixel outside above range
                img[i][j] = 0


    # convert back to BGR
    img = cv2.cvtColor(img, cv2.COLOR_HSV2BGR)
    return img

# import image
img = cv2.imread('selfie.png')

# detect face
img = detect_face(img)

# display image
cv2.imshow('image', img)
# wait for 3 seconds
cv2.waitKey(3000)
# close image
cv2.destroyAllWindows()

# save face
cv2.imwrite('face.png', img)
```

Figure 12: Skin Detection Python Code

2.3.2 Results

Figure 13: Original Image

Figure 14: Image outputted from skin detection