Supplementary material for "Research on the spectrum of smartphone screen based on laboratory-made spectrometer"

I. THEORY

A. Diffraction grating

We have chosen a reflective diffraction grating here, the diffracted light pattern is on the same side as the incident beam.

The dispersion that results from a diffraction grating is expressed in terms of the grating equation shown below, which relates the wavelength of the diffracted light to the diffraction and incident angles of the light that is directed from and to the grating and to the groove spacing:

$$n\lambda = d\left(\sin\theta_i + \sin\theta_d\right)$$

1) λ represents the wavelength of the diffracted light
2) d represents the groove spacing of the lines or grooves that are in the diffraction grating
3) θ_i is the incident angle of the wave that is directed towards the diffraction grating
4) θ_d is the diffraction angle of the light of wavelength λ that reflects off the grating
5) n is an integer value that designates the order of the diffraction

For polychromatic light, each individual wavelength will generate a diffraction pattern by satisfying the grating equation shown above for the specific value of the wavelength λ.

B. Detector width

The detector width is defined by three parameters: the bandwidth of the spectrometer $\Delta\lambda = \lambda_{max}$ - λ_{min}, the distance d of the grating slit, and the focal length f of the focusing lens. Among them, $\Delta\lambda$ and d are usually prerequisites, which can be selected to match the geometry of the detector by selecting the focusing lens.
Therefore, the width of the detector can be calculated using:

$$L = f\left[\tan\left(\beta_{max} - \alpha\right) + \tan\left(\beta_{min} - \alpha\right)\right]$$

where α is the incident angle and β_{max} and β_{min} are maximum and minimum diffraction angle respectively.

C. Specific calculation

In this spectrometer, if it is designed to analyze visible light in the wavelength range from $400nm$ to $700nm$, it is necessary to calculate the required detector width under the incident light conditions of the corresponding wavelengths.
We hope that the detector can collect the 1st-order diffracted light while avoiding the $0th - order$ diffracted light to obtain the clearest spectrum.

Therefore, $n = 1$ can be made, and the corresponding maximum and minimum diffraction angles can be calculated when the light of $400nm$ and $700nm$ respectively enters the fixed grating while keeping the incident angle constant.

Substituting the maximum and minimum diffraction angles into the detector width equation mentioned above allows the minimum required detector width to be calculated.

In this experiment, the incident angle is 25.6 degrees. When the wavelength is $400nm$, the corresponding diffraction angle is calculated to be 16.73 degrees. When the wavelength is $700nm$, the corresponding diffraction angle is 55.89 degrees. Therefore, it can be calculated that the required minimum detector width is $14.98mm$.

D. Blue light radiation ratio

In order to evaluate the blue light hazard degree of the light source under test, we need to calculate and evaluate the intensity of all blue light in the spectrum.

A concept is introduced here, called the blue light radiation ratio, which is defined as the ratio of the integral of the light intensity of the light whose wavelength belongs to the blue light part to the overall light intensity. The equation is represented as follows:

$$Ratio = \frac{\int_{\lambda=400nm}^{\lambda=480nm} I_{blue}}{I_{all}}$$

E. Color temperature

The color temperature calculation of the spectrum is based on the spectrogram data of the light source, and the corresponding color temperature is obtained by calculating its chromaticity coordinates (x, y) and performing interpolation. The following are the steps to calculate the color temperature of the spectrum:

1). According to the spectrogram data of the light source, calculate its tristimulus value XYZ.
2). Calculate the normalized tristimulus value XYZ, so that X+Y+Z=1.
3). Calculate the chromaticity coordinate xy, and normalize the tristimulus value XYZ into the plane, that is, $x = X/(X + Y + Z)$ and $y = Y/(X + Y + Z)$.
4). Calculate the corresponding chromaticity coordinates (xt, yt) according to the standard blackbody radiation, where t is the blackbody temperature.
5). Take the distance between the chromaticity coordinates (xt, yt) and the chromaticity coordinates (x, y) as the chromatic aberration ΔE_{ab}, and then use interpolation to find the black body temperature that minimizes the chromatic aberration, which is the spectral color temperature sought.

The calculation of the color temperature of the spectrum depends on many factors, such as the spectral distribution of the light source, the standard black body radiation spectrum, etc., and the calculation complexity is relatively high. In this experiment, Python software is used to realize the above calculations.

F. Color difference

The principle of calculating color difference is based on the perception of color by the human visual system. The color difference is a measure of the difference between two colors and is commonly used in areas such as color management, color matching, and quality control. When calculating color difference, people usually use different color spaces and color difference formulas. One of the most commonly used color spaces is the CIE Lab color space, which has a good correspondence: a color space in which the distance between two points is roughly proportional to the human perception of color differences.

The CIEDE2000 color difference formula is one of the popular color difference formulas, which is an improvement based on the CIE Lab color space. CIEDE2000 takes into account the non-uniform perception of hue, saturation, and brightness by the human visual system. Compared with the traditional CIE76 and CIE94 color difference formulas, CIEDE2000 is closer to a human's actual perception of color difference.

The calculation of the CIEDE2000 color difference formula involves multiple steps and correction items. The following is an overview of the CIEDE2000 color difference calculation:

1). First, in the CIE Lab color space, calculate the lightness (L), chromaticity (a and b), and hue angle of two colors.
2). Correct brightness, chroma, and hue. Corrections include:

I. Symmetry adjustment for color difference.
II. Hue Continuity Adjustment.

III. Hue-dependent brightness adjustment.
IV. Hue-dependent chroma adjustment.
3). Calculate the corrected hue difference, lightness difference, and chromaticity difference.
4). Calculate the result of the CIEDE2000 color difference formula based on the corrected difference.

The detailed calculation steps of the CIEDE2000 color difference formula are relatively complicated, but the existing color processing library (such as the color math library of Python) has realized these calculations. In practical applications, we can directly use these libraries to calculate CIEDE2000 color differences without manually implementing complex mathematical formulas.

II. EXPERIMENT DESIGN

A. Experiment Real Set-up

Firstly, use an aperture for generating a light dot. A one-direction light dot in an optical system is of great benefit: it ensures that the light entering the system has a well-defined spatial distribution, leading to better resolution and reduced stray light. We choose an aperture instead of a slit or optical fiber because apertures provide more control over the shape and size of the light dot, allowing for better alignment and customization of the optical setup.

Then, the light will go through a 50mm plano-convex spherical lens for calibration. We need a collimated light in the system as it helps minimize aberrations and distortions, ensuring that the light rays are parallel and maintaining the spatial coherence of the beam.

After that, the collimated light will shoot on the grating and be diffracted. As the light will be diffracted in many directions, we need to select the appropriate order of the diffracted light to obtain the highest resolution and signal-to-noise ratio for the spectrum.

However, we need another lens to focus the light band since the light band is hard to capture by the camera. This is because, without focusing, the image will be too large, and a web camera is more capable of capturing the image shown on the surface of the lens when the light band is focused onto a smaller area. Here we use a 35mm plano-convex spherical lens.

At the same time, we added a mirror between the detector and the focusing lens, so that the camera can receive the spectrum reflected from the mirror, which is necessary for the spatial arrangement of the entire experimental equipment. By incorporating a mirror into our setup, we were able to deviate from the theoretical configuration and achieve a more practical and optimized placement for the camera.

Finally, the camera captures the spectrum, and we can use software to analyze it. This step involves processing the captured image data, extracting the relevant spectral information, and converting it into a format that can be further analyzed or visualized.

The equipment used is listed below. Generally, the size of all the lenses and mirror are: $12.7mm \times 12.7mm$.
1). Input slit: aperture
2). Collimating lens: 50mm Plano-Convex Spherical Lens
3). Grating: reflecting grating, model: 1800 grooves/mm
4). Focusing lens: 35mm Plano-Convex Spherical Lens
5). Mirror
6). Web camera with 1920×1080 pixels
7). Opaque-to-light cardboard box

B. Experimental process for setting up the optical system

1. Pre-setup with high-intensity flash light

Begin by using a high-intensity flash light as the light source for the initial setup. This will help to visualize the optical components' positions and alignments. Adjust the positions of the aperture, lenses, and grating to create a

clear and well-defined light path.

2. Screen light test and fine adjustments for final setup

After the pre-setup, use screen light to further test the optical path. This will provide more precise information about the system's performance. Carefully adjust the positions and angles of the components to optimize the collimation, focusing, and diffraction of the light. Once the fine adjustments are made, the final setup will be complete.

3. Cover the setup and perform testing

To minimize the influence of external light sources and ensure accurate measurements, cover the entire setup with a box or enclosure. This will create a controlled environment for the optical system. Moving forward, we will conduct tests on the system, capturing the spectrum with the web camera and analyzing the results using suitable software.: Theremino Spectrometer.

We utilized spectral analysis software developed by Theremino to acquire and evaluate spectra. The software can determine the intensity of light received by each pixel and subsequently measure the radiation levels for each color. It is worth noting that the camera's ability to distinguish photon colors is not relevant; rather, the intensity and location of the photons are the key factors. A monochrome camera would suffice as red light only hits certain pixels, green light hits different pixels, and violet, infrared, or other colors hit distinct pixels.

However, it is important to acknowledge the limitations of using a webcam-based spectrometer. The software is optimized for a spectrometer based on a webcam, and it can only measure wavelengths, not the quantity of light. Furthermore, all cameras, including webcams, have a non-linear response, making it impossible to conduct precise quantitative measurements. As a result, only relative and approximate differences can be discerned, and the amount of light cannot be accurately measured.

III. EXPERIMENT PROCESS

A. Calibration

Various methods have been explored for calibration from different perspectives:

Regarding light path calibration, collimation has been achieved using a cylindrical collimator, slit, and diverse lenses. Concurrently, the height and distance of each optical apparatus have been adjusted to ensure that light is imaged at the center of each lens, thereby obtaining the cleanest and most comprehensive light spectrum from the camera. Ultimately, a slit and hole on a box plate were employed in place of the cylindrical collimator for a stable light source.

Concerning the light wavelength calibration of the Theremino Spectrometer-V3.1 software, a standard wavelength range was established for the detected spectrum, which was tested using various colors of screen light. The standard wavelength values for the software were set based on the OLED screen's wavelength range and the software's color performance, with blue at 440 nm, green at $550nm$, and red at $660nm$.

Use white screen light to do wavelength calibration, and the corresponding result from the software is showed in Figure S1.

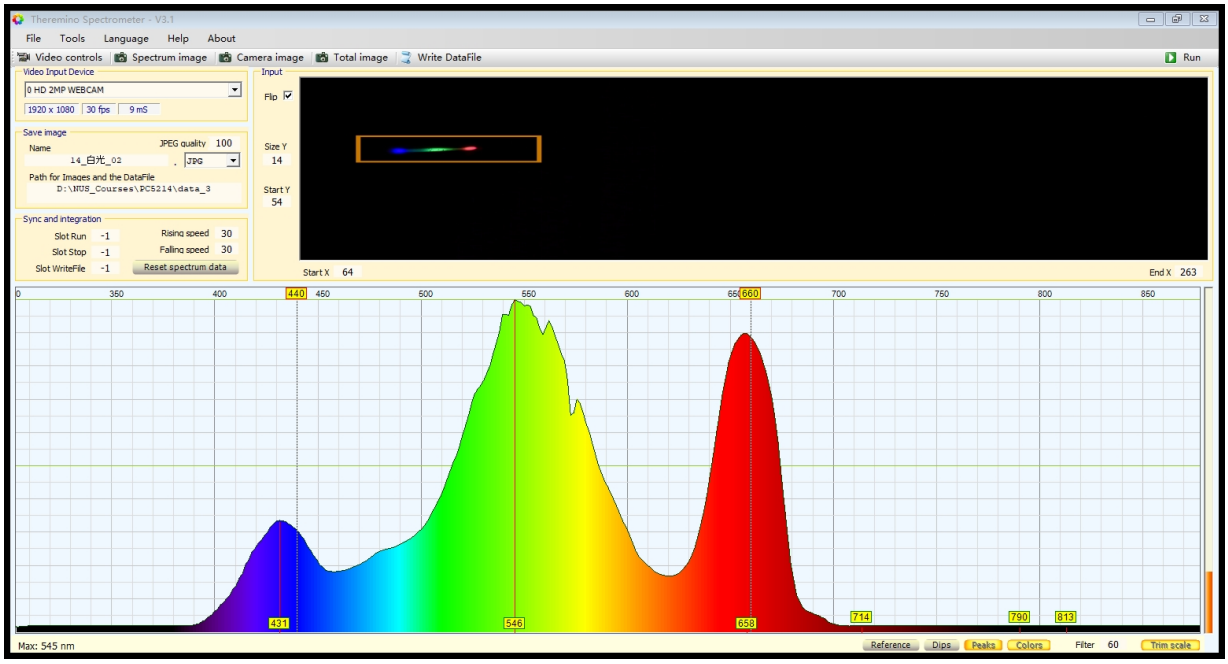

FIG. S1. Calibrate the software, showing the result when the white screen light is injected into the system.

Here we detect the wavelength of peak value are 431nm, 546nm and 658nm for blue, green and red respectively. This can prove our setting values are accurate for wavelength.The intensity of blue light is smaller than two other colors because limitation of lens (focal length or diameter) which causes spectrum can't be imaged comprehensively and especially for blue light spectrum be formed on the edge or extended the edge of lens. Therefore, we did trade-off between completion and resolution of spectrum.

B. Light source preparation

Three colors (green, blue, red) and white of screen light from iPhone 11, 12, 13, and 14 were utilized as light sources for comparative analysis.

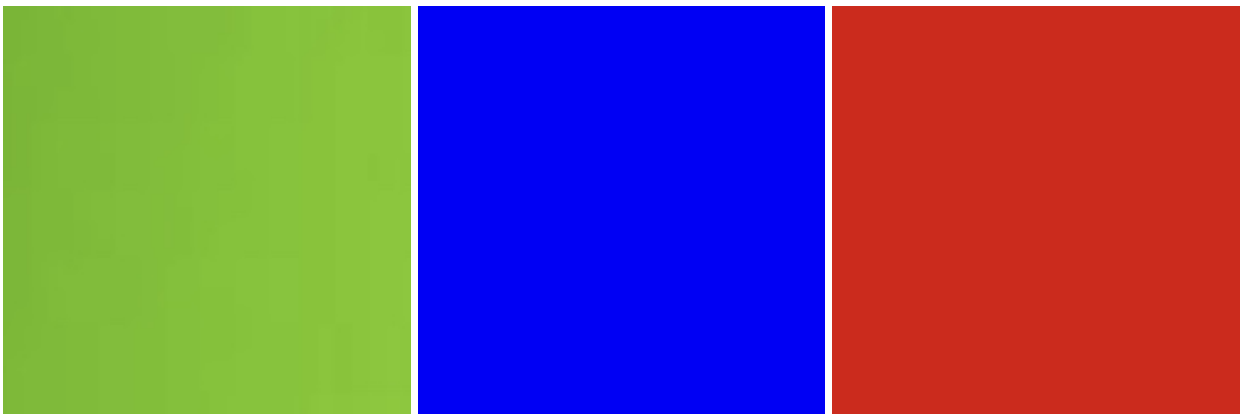

FIG. S2. Three different color input.

1).Download the pure and same color images from website.
2).Fix the brightness and displayed time of each phone is maximized to prevent some noise from screen manipulation.
3).Make every color image to extend the whole iPhone screen without margin.
4).Shut off all light source in the room and draw curtains to create dark environment for improving contrast of image.

C. Spectrometer simulating process

1).Put screen with slit tightly and cover a box on the whole set up to ensure no more external light spread into environment for obtaining better quality of spectrum images.
2).Light goes through the hole and lens, then collimate light hits grating to form spectrum.
3).The spectrum image enters focus lens and reflected by mirror lens and captured by camera.
4).The camera connects to laptop, and we can see the spectrum image and some analysis from software (Theremino Spectrometer-V3.1)

D. Acquisition of experimental data

All the results obtained from the experiment are shown here in Figure S3.

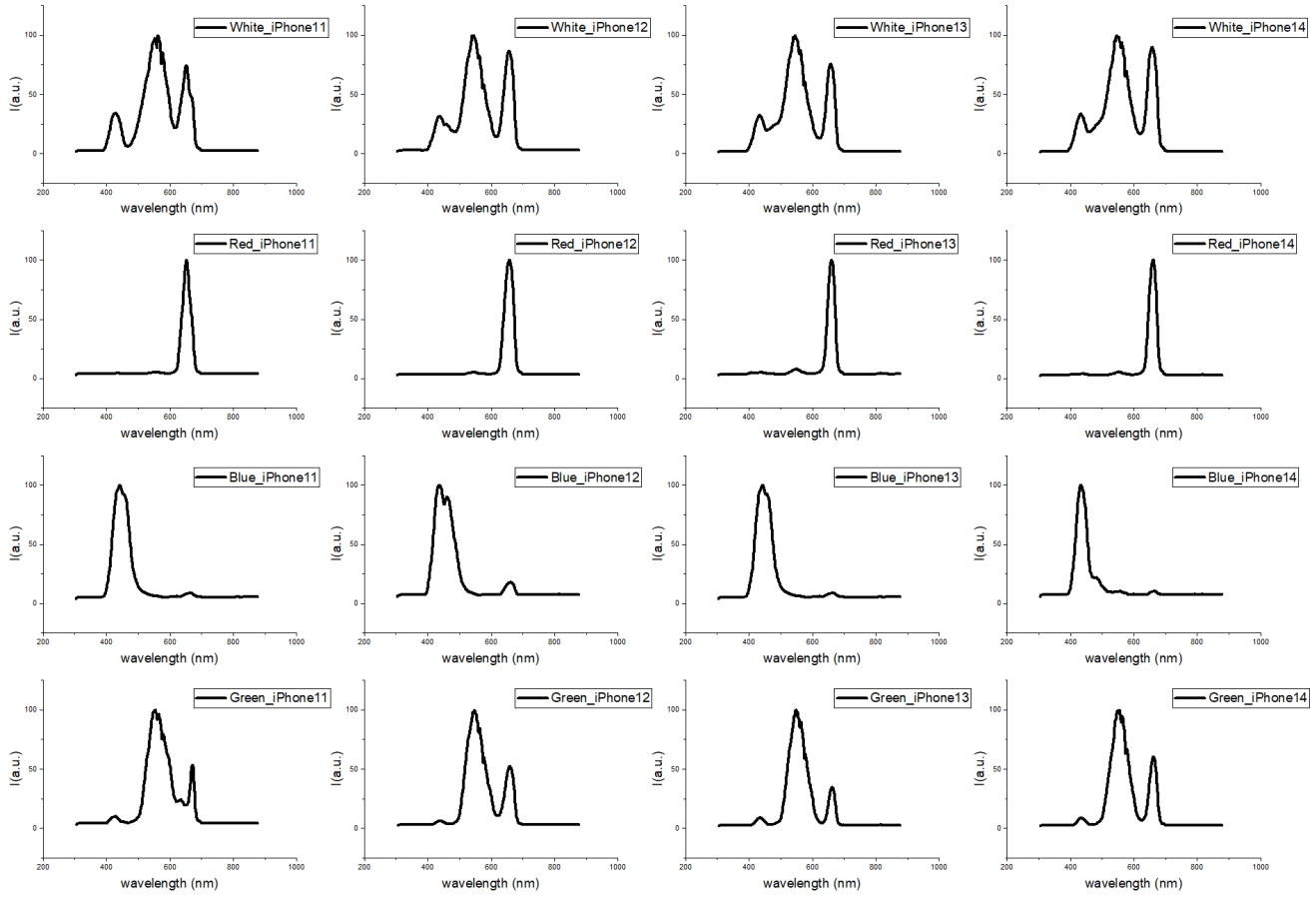

FIG. S3. All the results obtained from the experiment, the order on the column is white light, red light, blue light, green light. The order on the row is iPhone 11 to 14.

E. Python code of Blue Light Radiation Ratio

```
import numpy as np
import pandas as pd

def calculate_blue_light_percent(spectrum):
```

```
    blue_light_range = (spectrum[:, 0] >= 450) & (spectrum[:, 0] <= 495)
    blue_light_spectrum = spectrum[blue_light_range]
    blue_light_intensity_sum = np.sum(blue_light_spectrum[:, 1])
    total_intensity_sum = np.sum(spectrum[:, 1])
    blue_light_percentage = blue_light_intensity_sum / total_intensity_sum
    return blue_light_percentage

spectrum = []
blue_light_percent = []
for i in range(11, 15):
    dataset_filename = f"{i}_white.txt"
    spectrum_i = np.loadtxt(dataset_filename, skiprows=2)
    blue_light_percent_i = calculate_blue_light_percent(spectrum_i)
    spectrum.append(f"iphone␣{i}")
    blue_light_percent.append(blue_light_percent_i)

table = pd.DataFrame(data={"Spectrum": spectrum, "Blue␣light␣percent": blue_light_percent})
print(table)
```

F. Python code of color temperate

```
import numpy as np
import pandas as pd
from scipy.optimize import minimize_scalar
from scipy.integrate import simps


wavelength_range = [360,830]
def load_cmf(filename):
    data = pd.read_csv(filename, header=None, names=['wavelength', 'x', 'y', 'z'])
    cmf_wavelengths = data['wavelength'].values
    cmf_indices = np.where((cmf_wavelengths >= wavelength_range[0]) & (cmf_wavelengths <=
        wavelength_range[1]))[0]
    cmf_wavelengths = cmf_wavelengths[cmf_indices]
    x = np.interp(cmf_wavelengths, data['wavelength'].values, data['x'].values)
    y = np.interp(cmf_wavelengths, data['wavelength'].values, data['y'].values)
    z = np.interp(cmf_wavelengths, data['wavelength'].values, data['z'].values)
    cmf = {
        'wavelength': cmf_wavelengths,
        'x': x,
        'y': y,
        'z': z,
    }
    return cmf

def load_spd(filename, cmf):
    data = pd.read_csv(filename, sep='\t', skiprows=2, header=None, names=['wavelength', '
        intensity'])
    spd_wavelengths = np.round(data['wavelength']).astype(int).values
    spd_indices = np.where((spd_wavelengths >= wavelength_range[0]) & (spd_wavelengths <=
        wavelength_range[1]))[0]
    spd_wavelengths = spd_wavelengths[spd_indices]
    spd_intensities = data['intensity'].values[spd_indices]
    cmf_wavelengths = cmf['wavelength']
    x = np.interp(cmf_wavelengths, spd_wavelengths, spd_intensities)
```

```
    spd = {
        'wavelength': cmf_wavelengths,
        'intensity': x,
    }
    return spd

def spd_to_xyz(spd, wavelength, cmf):
    X = simps(spd * cmf['x'], wavelength)
    Y = simps(spd * cmf['y'], wavelength)
    Z = simps(spd * cmf['z'], wavelength)
    return np.array([X, Y, Z])

def planckian_locus(T, wavelength):
    h = 6.62607004e-34
    c = 3.0e8
    k = 1.38064852e-23
    nm_to_m = 1e-9
    T = T * np.ones_like(wavelength)
    B = (2 * h * c ** 2) / (wavelength * nm_to_m) ** 5 * (1 / (np.exp((h * c) / (wavelength *
        nm_to_m * k * T)) - 1))
    return B

def color_temperature(spd, wavelength, cmf):
    xyz = spd_to_xyz(spd, wavelength, cmf)
    x = xyz[0] / np.sum(xyz)
    y = xyz[1] / np.sum(xyz)

    def objective_function(T):
        B = planckian_locus(T, wavelength)
        xyz_blackbody = spd_to_xyz(B, wavelength, cmf)
        x_blackbody = xyz_blackbody[0] / np.sum(xyz_blackbody)
        y_blackbody = xyz_blackbody[1] / np.sum(xyz_blackbody)
        return (x - x_blackbody) ** 2 + (y - y_blackbody) ** 2

    result = minimize_scalar(objective_function, bounds=(1000, 100000), method='bounded')
    return result.x

# CIE 1931 2-deg Standard Observer Color Matching Functions (CMFs)
# Load CMF data
cmf_data = load_cmf('CIE_cc_1931_2deg.csv')

# Load SPD data
# spd_data = load_spd("14_white.txt",cmf_data)

# ct = color_temperature(spd_data['intensity'], spd_data['wavelength'] , cmf_data)
# print(f"Color temperature: {ct:.2f} K")

phone = []
color_temp = []

for i in range (11,15):
    dataset_filename = f"{i}_white.txt"
    # Load SPD data
    spd_data = load_spd(dataset_filename,cmf_data)
    ### calculate the color temperature
    ct = color_temperature(spd_data['intensity'], spd_data['wavelength'] , cmf_data)
    color_temp.append(ct)
    phone.append(f"iphone␣{i}")
```

```
table = pd.DataFrame(data={"Phone": phone, "Color␣temperature": color_temp})
print(table)
```

G. Python code of color difference

```
from colormath.color_objects import LabColor, XYZColor
from colormath.color_conversions import convert_color
from colormath.color_diff import delta_e_cie2000
from scipy.integrate import simps
import pandas as pd
import numpy as np

wavelength_range = [360,830]
def load_cmf(filename):
    data = pd.read_csv(filename, header=None, names=['wavelength', 'x', 'y', 'z'])
    cmf_wavelengths = data['wavelength'].values
    cmf_indices = np.where((cmf_wavelengths >= wavelength_range[0]) & (cmf_wavelengths <=
        wavelength_range[1]))[0]
    cmf_wavelengths = cmf_wavelengths[cmf_indices]
    x = np.interp(cmf_wavelengths, data['wavelength'].values, data['x'].values)
    y = np.interp(cmf_wavelengths, data['wavelength'].values, data['y'].values)
    z = np.interp(cmf_wavelengths, data['wavelength'].values, data['z'].values)
    cmf = {
        'wavelength': cmf_wavelengths,
        'x': x,
        'y': y,
        'z': z,
    }
    return cmf

def load_spd(filename, cmf):
    data = pd.read_csv(filename, sep='\t', skiprows=2, header=None, names=['wavelength', '
        intensity'])
    spd_wavelengths = np.round(data['wavelength']).astype(int).values
    spd_indices = np.where((spd_wavelengths >= wavelength_range[0]) & (spd_wavelengths <=
        wavelength_range[1]))[0]
    spd_wavelengths = spd_wavelengths[spd_indices]
    spd_intensities = data['intensity'].values[spd_indices]
    cmf_wavelengths = cmf['wavelength']
    x = np.interp(cmf_wavelengths, spd_wavelengths, spd_intensities)

    spd = {
        'wavelength': cmf_wavelengths,
        'intensity': x,
    }
    return spd

def spd_to_xyz(spd, wavelength, cmf):
    X = simps(spd * cmf['x'], wavelength)
    Y = simps(spd * cmf['y'], wavelength)
    Z = simps(spd * cmf['z'], wavelength)
    return np.array([X, Y, Z])


def xyz_to_lab(xyz):
```

```
    xyz_color = XYZColor(*xyz)
    lab_color = convert_color(xyz_color, LabColor)
    return lab_color

def color_difference(filename1, filename2, cmf_data):
    spd_data1 = load_spd(filename1, cmf_data)
    spd_data2 = load_spd(filename2, cmf_data)

    xyz1 = spd_to_xyz(spd_data1['intensity'], spd_data1['wavelength'], cmf_data)
    xyz2 = spd_to_xyz(spd_data2['intensity'], spd_data2['wavelength'], cmf_data)

    lab1 = xyz_to_lab(xyz1)
    lab2 = xyz_to_lab(xyz2)

    color_diff = delta_e_cie2000(lab1, lab2)
    return color_diff



iphone_model = []
color_diff_set = []
# calculate the color difference
cmf_data = load_cmf('CIE_cc_1931_2deg.csv')

for i in range (11,15):
    if i <= 13:
        dataset_filename_1 = f"{i}_white.txt"
        dataset_filename_2 = f"{i+1}_white.txt"
        color_diff = color_difference(dataset_filename_1, dataset_filename_2, cmf_data)
        iphone_model.append(f"iPhone {i} - iPhone {i+1}")
        color_diff_set.append(color_diff)
        if i<= 12:
            dataset_filename_1 = f"{i}_white.txt"
            dataset_filename_2 = f"{i+2}_white.txt"
            color_diff = color_difference(dataset_filename_1, dataset_filename_2, cmf_data)
            iphone_model.append(f"iPhone {i} - iPhone {i+2}")
            color_diff_set.append(color_diff)
            if i<= 11:
                dataset_filename_1 = f"{i}_white.txt"
                dataset_filename_2 = f"{i+3}_white.txt"
                color_diff = color_difference(dataset_filename_1, dataset_filename_2, cmf_data)
                iphone_model.append(f"iPhone {i} - iPhone {i+3}")
                color_diff_set.append(color_diff)

table = pd.DataFrame(data={"Phone Model": iphone_model, "Color Difference (%)": color_diff_set})
print(table)
```