Team:16
Version:1
Date:02/06/2023

Standard Operating Procedure

I. **Protocol Title:** Code Development and Maintenance

II. **Brief Description of Protocol:**

This SOP outlines the procedure for developing and maintaining the code used for data analysis in the project. It covers the coding standards, best techniques/practices, testing guidelines, and error handling. It ensures consistency, readability, and maintainability of the codebase. Proper variable naming conventions, coding techniques, code organization, and documentation are provided in the SOP. Emphasizes the use of reusable code components to promote efficiency and scalability. Encourages the use of inline comments for comprehensive documentation to provide clarity and context to the codebase. It outlines the function descriptions, and input/output specification. Promotes standardized code development and maintenance by fostering quality, reliability, and collaboration in this project.

III. **Materials**

- Test strip images saturated with modified Scotts Reagent
- Alcohol Solution containing Benedryl (substitute for ketamine)
- Smartphone Camera or imaging device for capturing test strip images
- Anaconda: Python programming environment
- Jupyter Notebook: Data Analysis and Visualization
- Required Python Libraries: os, cv2, numpy, matplotlib.pyplot
- GitHub portfolio and repository for Senior Design Project
- GitHub Link: GitHub Link for Code Development and Maintenance

IV. **Procedure**

1. **Coding Standards, Best Practices, and Code Documents:**

Note: All the comments in the code snippets below are for clarity and to help readers understand the code's functionality.

```
import os
#Change the current working directory to the specific path
os.chdir('C:/Users/user/Pictures/BIEN175B')
#Get the current working directory and print it
current_dir=os.getcwd()
print("Current working directory:", current_dir)
```

The above code snippet allows to run commands in the Python script. Includes information about changing the current directory using 'os.chdir()' and retrieving the current working directory using 'os.getcwd()'. It gives access to the specific location where all the test strip images are stored and that will be used for data analysis.

Team:16
Version:1
Date:02/06/2023

```python
import cv2 #import Computer Vision library

# Load the images
img0 = cv2.imread('blue_img0.png')
img1 = cv2.imread('blue_img1.png')
img2 = cv2.imread('blue_img2.png')
img3 = cv2.imread('blue_img3.png')

import numpy as np #import numerical python library for calculation

#Calculate the Average Pixel Values for color channels in img0,img1,img2,img3
mean0=cv2.mean(img0)
mean0=[round(val,2) for val in mean0[:3]]

mean1=cv2.mean(img1)
mean1=[round(val,2) for val in mean1[:3]]

mean2=cv2.mean(img2)
mean2=[round(val,2) for val in mean2[:3]]

mean3=cv2.mean(img3)
mean3=[round(val,2) for val in mean3[:3]]
```

```python
#Print the average vlaues for each image
print("Below we print the average pixel values of R, G, B component in image-1")
print(mean0[:3])

print("Below we print the average pixel values of R, G, B component in image-2")
print(mean1[:3])

print("Below we print the average pixel values of R, G, B component in image-3")
print(mean2[:3])

print("Below we print the average pixel values of R, G, B component in image-4")
print(mean3[:3])
```

The above code snippets are documented by adding comments explaining each section's purpose and functionality. It includes loading the images, calculating the average pixel values for each color component channel, and finally printing the results.

Team:16
Version:1
Date:02/06/2023

```
import numpy as np #imported numerical python library for calculation
import matplotlib.pyplot as plt #imported for interactive plots generation

# Define the data
labels = ['Simply Spiked(5%)', 'Smirnoff(4.5%)', 'Cayman Jack(5.8%)', 'Cayman Jack(5.8%)\n+Black Dye']
#Subtract the obtained RGB values by 255. This will give higher RGB vlaues for darker colors
red_means = [255-207.01,255-204.42, 255-194.21, 255-169.87]
green_means = [255-201.64,255-201.83, 255-194.35, 255-165.69]
blue_means = [255-177.72,255-184.44, 255-178.62, 255-150.87]

# Set the positions and width of the bars
pos = np.arange(len(labels))
width = 0.25
offset=0.12

# Create the figure and axis objects
fig, ax = plt.subplots(figsize=(10, 6))

# Create the bar plots
rects1 = ax.bar(pos, red_means, width, color='r', alpha=0.5)
rects2 = ax.bar(pos + width, green_means, width, color='g', alpha=0.5)
rects3 = ax.bar(pos + 2 * width, blue_means, width, color='b', alpha=0.5)

# Set the axis labels and title
ax.set_xlabel('Test Strip Samples of Hard Seltzers with \u22482% Diphenhydramine HCl (Benadryl)')
ax.set_ylabel('RGB Color Intensity (Pixels)')
ax.set_title('RGB Color Intensity Analysis of Test Strip Samples')
```

```
# Set the axis ticks and tick labels
ax.set_xticks(pos + width * 1.5-offset)
ax.set_xticklabels(labels,  ha='center')
ax.set_yticks(np.arange(0, 256, 25))

# Add the legend
ax.legend((rects1[0], rects2[0], rects3[0]), ('Red', 'Green', 'Blue'))

# Display the plot
plt.show()
```

The above code snippet generates a bar plot using the ‘matplotlib.pyplot’ library to visualize the RGB color intensity values for different test strip samples. This generates three sets of bars representing red, green, and blue color channels. The x-axis displays the labels of the test strip samples, while the y-axis represents RGB color intensity in pixels. The color intensity values are calculated by subtracting the obtained RGB values from 255, resulting in much higher RGB values for darker colors. The plot includes a legend indicating the colors that are associated with each set of bars. This visual representation assists in comparing and analyzing the RGB color intensities of test strip samples.

All the above code blocks can be used for various types of comparison between different cases of test strips with a few changes block by block. Some of these are Comparisons of RGB Values for TEST STRIP with Reagent, to observe the change in RGB values for loss of water molecules, to observe Changes in TEST STRIP after 19 DAYS, to perform RGB comparison of NEGATIVE CONTROL (i.e in Water), RGB Color Comparison of BENEDRYL in KOMBUCHA, Comparison of RGB values for Loss of Water Molecules, etc.

So the above code blocks can be reused for different test strip cases to calculate RGB values, and generate informative bar graphs by making individualized changes based on test cases.

2. **Version Control:**
   The GitHub link below has all the versions of code for different test cases mentioned in the previous paragraph above
   GitHub Link: [GitHub Link for Code Development and Maintenance]

- This Version control system tracks all the code changes made while doing the project.
- Repositories to store the code can be used by teammates and for future collaboration.
- Different techniques used are also mentioned.
- Documents any updates or enhancements made to the codebase.

3. Error Management:

- Error handling is effectively done by implementing mechanisms such as defining variables and adding checking points by using print states such as “Read”. Additionally, error logging and error messages helped in diagnosing, and troubleshooting issues, facilitating efficient debugging and maintenance of the codebase.

Team:16
Version:1
Date:02/06/2023

V. References:

1. Prasannahariveeresh, “Implementing Lane Detection in OpenCV Python: A Step-by-Step Beginner’s Guide - Prasannahariveeresh.” *Prasannahariveeresh - Read My Tech Blogs Here*, 18 Jan. 2023, https://jrprasanna.com/2023/01/18/implementing-lane-detection-in-opencv-python-a-step-by-step-beginners-guide/
2. “Matplotlib Pie Charts.” *Vegibit*, vegibit.com/matplotlib-pie-charts/. Accessed 19 May 2023. https://vegibit.com/matplotlib-pie-charts/