Software Team of Skyxperts

Task 2

Name:Mohamed Zakaria

1) Description

These project is an application that's based on PyQt for color space editing using OpenCV and, consists of main window with several sliders and an "EXIT" button. By setting the sliders, the user can change the color ranges and see the effects in real-time through the webcam stream.

The main features of the program include:

- The application window is created using PyQt, with a layout containing multiple horizontal slider layouts, a label for each slider, and an "Exit" button.
- The sliders allow the user to adjust various parameters related to color space editing, such as hue, saturation, and value.
- The current values of the sliders are displayed dynamically next to each slider using QLabel widgets.
- The webcam stream is captured using the OpenCV library and processed in real-time based on the user's slider adjustments.
- The processed frame is displayed in two windows: "Webcam Stream" and "result". The "result" window shows the frame with the applied color space filtering based on the slider values.
- The application continues to process frames from the webcam until the user clicks the "Exit" button or presses the 'q' key.

To conclude, these application provides a basic interface for experimenting with color space editing using OpenCV and PyQt.

2) Requirments file

1) PyQt6: Library used for creating the GUI interface.

Those steps for how to install the Wheel for the GPL version of PyQt6:

- pip install PyQt6
- pip install PyQt-builder

2) OpenCV: Used for capturing webcam frames and image processing.

Steps to install OpenCV:

- pip install opencv-python

3) numpy: Required by OpenCV for numerical operations.

The steps to install numpy:

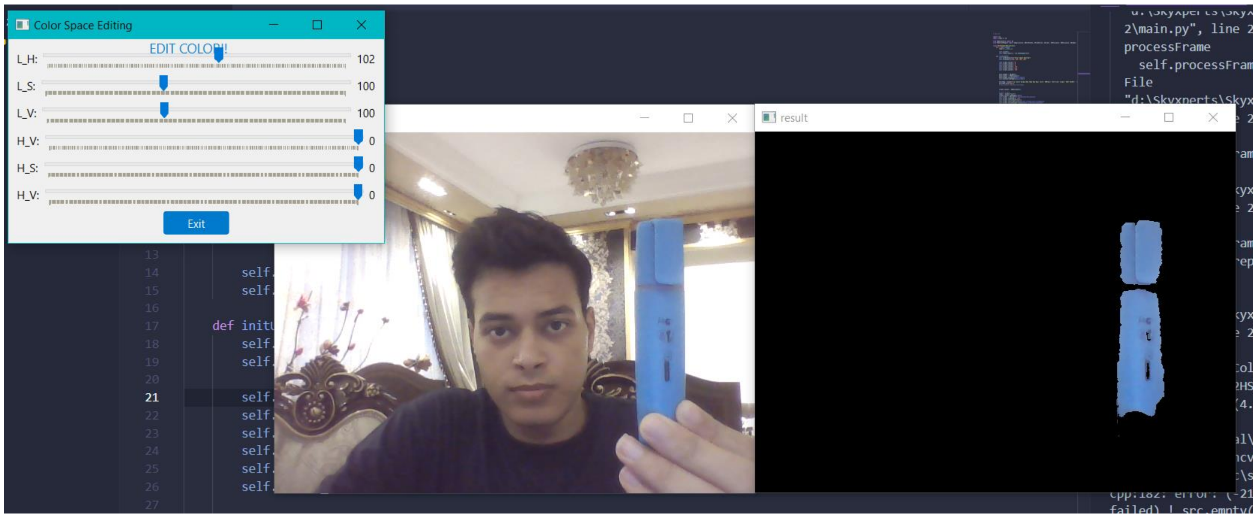
- pip install numpy

3) to run the code:

➔ type in the terminal python main.py

4) The output:

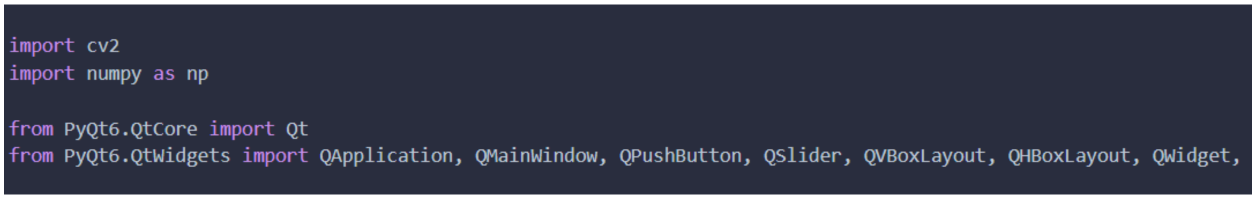
I Adjust the track bars to the values belongs to blue color and this is the outcome that give it me!



L_H: 102
L_S: 100
L_V: 100
H_V: 0
H_S: 0
H_V: 0

5) The code flow:

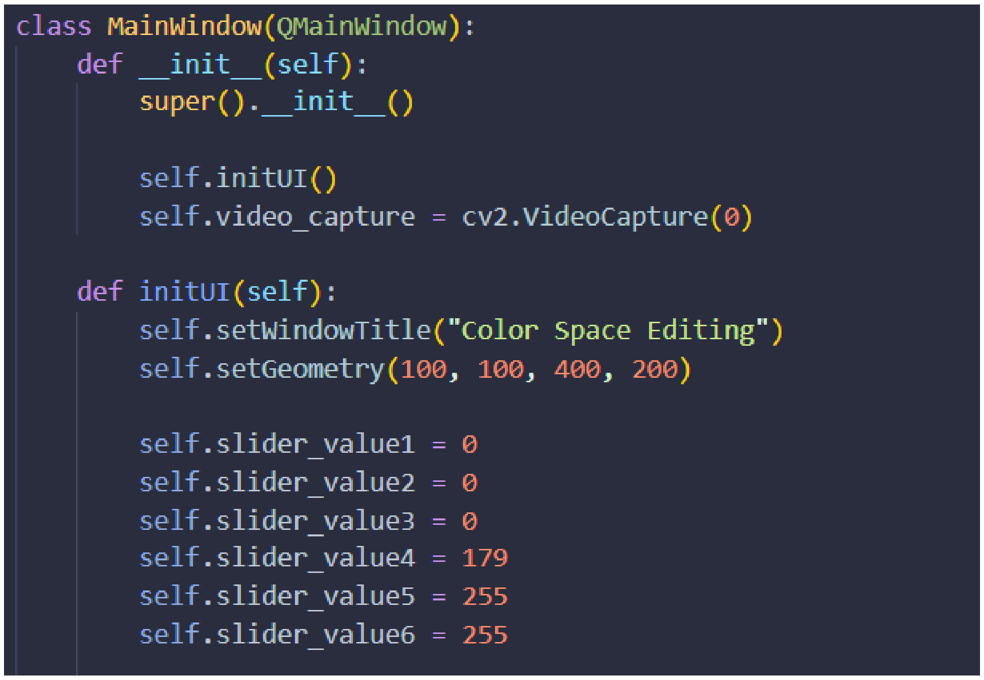
1) The necessary imports are made. The cv2 module is imported from OpenCV, and various classes and modules are imported from PyQt6.



```
import cv2
import numpy as np

from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton, QSlider, QVBoxLayout, QHBoxLayout, QWidget,
```

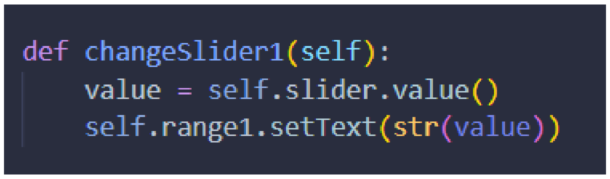
2) The MainWindow class is defined, which inherits from QMainWindow. The __init__ method initializes the main window and sets up the UI. It also initializes the video_capture object to capture video from the webcam.

3) The initUI method is defined to set up the user interface. It creates and configures several sliders, labels, and a button using PyQt6's layout system.



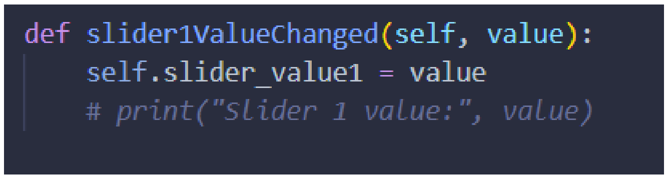
```python
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.initUI()
        self.video_capture = cv2.VideoCapture(0)

    def initUI(self):
        self.setWindowTitle("Color Space Editing")
        self.setGeometry(100, 100, 400, 200)

        self.slider_value1 = 0
        self.slider_value2 = 0
        self.slider_value3 = 0
        self.slider_value4 = 179
        self.slider_value5 = 255
        self.slider_value6 = 255
```

4) The changeSlider methods are defined to update the corresponding label's text when the sliders' values change.



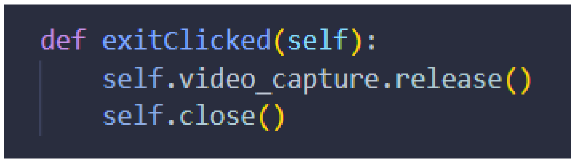
```python
def changeSlider1(self):
    value = self.slider.value()
    self.range1.setText(str(value))
```

5) The sliderValueChanged methods are defined to update the instance variables (slider_value1 through slider_value6) when the sliders' values change.



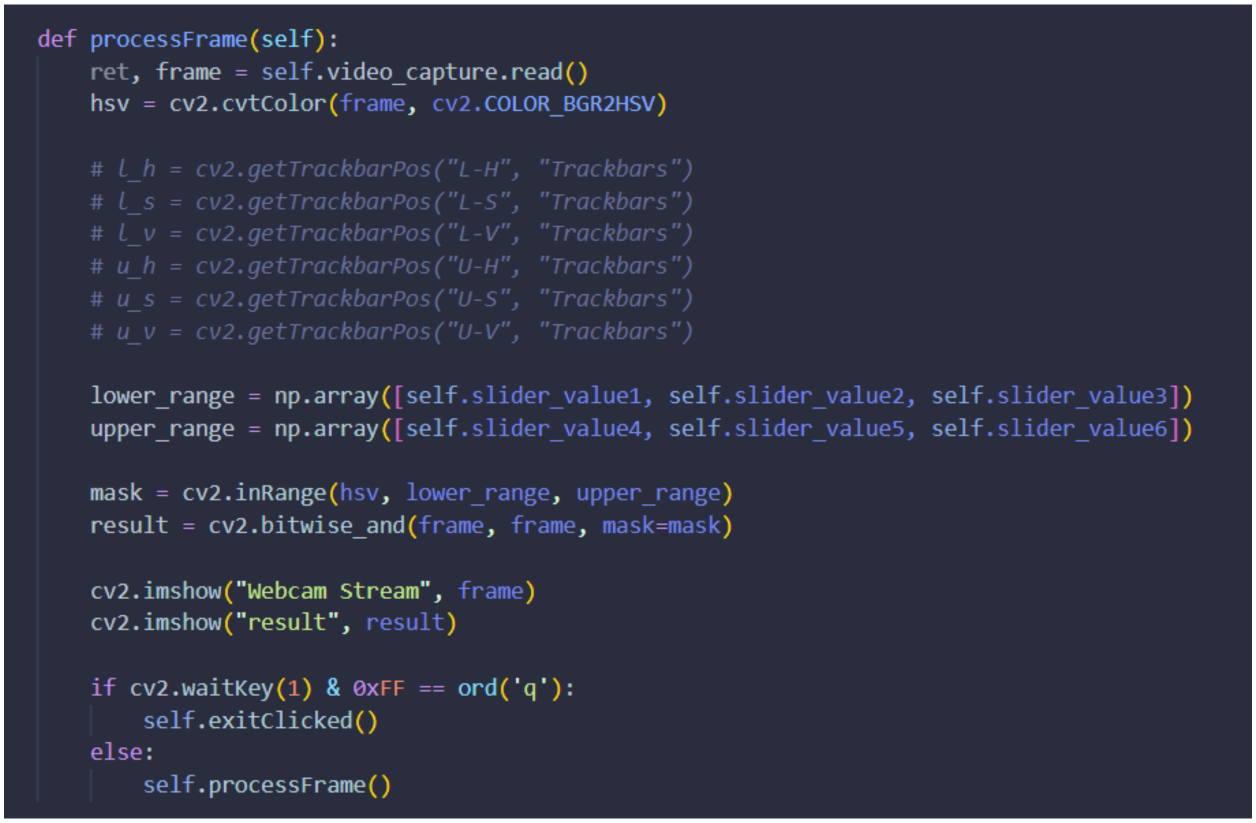
```python
def slider1ValueChanged(self, value):
    self.slider_value1 = value
    # print("Slider 1 value:", value)
```

6) The exitClicked method is defined to release the video capture object and close the application when the "Exit" button is clicked.



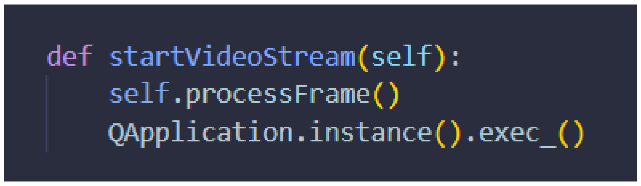
```python
def exitClicked(self):
    self.video_capture.release()
    self.close()
```

7) The processFrame method is defined to read frames from the video capture object, convert them to the HSV color space, apply a color range mask based on the slider values, and display the original frame and the result frame in separate windows. It also checks for the "q" key press to exit the application.



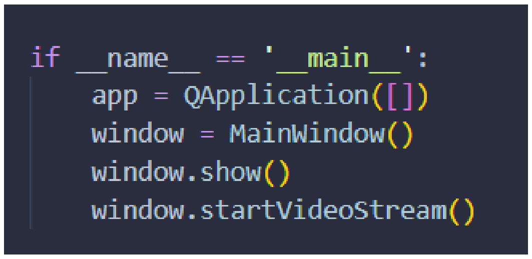
```python
def processFrame(self):
    ret, frame = self.video_capture.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # l_h = cv2.getTrackbarPos("L-H", "Trackbars")
    # l_s = cv2.getTrackbarPos("L-S", "Trackbars")
    # l_v = cv2.getTrackbarPos("L-V", "Trackbars")
    # u_h = cv2.getTrackbarPos("U-H", "Trackbars")
    # u_s = cv2.getTrackbarPos("U-S", "Trackbars")
    # u_v = cv2.getTrackbarPos("U-V", "Trackbars")

    lower_range = np.array([self.slider_value1, self.slider_value2, self.slider_value3])
    upper_range = np.array([self.slider_value4, self.slider_value5, self.slider_value6])

    mask = cv2.inRange(hsv, lower_range, upper_range)
    result = cv2.bitwise_and(frame, frame, mask=mask)

    cv2.imshow("Webcam Stream", frame)
    cv2.imshow("result", result)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        self.exitClicked()
    else:
        self.processFrame()
```

8) The startVideoStream method is defined to call the processFrame method in a loop and start the GUI event loop using QApplication.instance().exec_().



```python
def startVideoStream(self):
    self.processFrame()
    QApplication.instance().exec_()
```

9) The __main__ block creates an instance of the QApplication class, creates an instance of the MainWindow class, shows the main window, and starts the video stream by calling the startVideoStream method.



```python
if __name__ == '__main__':
    app = QApplication([])
    window = MainWindow()
    window.show()
    window.startVideoStream()
```