CS5330 - pattern Recognition and Computer Vision - Project 1
Authored by - Aadhi Aadhavan Balasubramanian & Harishraj Udaya Bhaskar

1) A short description of the overall project in your own words. (200 words or less)

The project utilizes the OpenCV library to access and process video from a camera source in real time. It commences by initializing a video capture. Following the initialization, the program creates a graphical window where the processed video stream will be displayed, allowing users to visualize the real-time filters being applied. To facilitate user interaction, the code defines several keystrokes corresponding to different image filters. There is a wide range of filters, including grayscale, sepia, blur, edge detection, sobel_x, solel_y, gradient, and more. The different keystrokes enable users to switch between different filter options during the program's execution.

The program continuously captures frames from the camera source. Inside this loop, the selected image processing filter is applied to each frame based on the keystroke given by the user. Users can trigger the program to switch between filters by pressing specific keys, allowing them to observe how different filters affect the live video stream. Furthermore, the program is designed to be user-friendly and interactive. Users can save frames as images. This feature is particularly useful for analysis or documentation purposes.

Finally, when the user decides to exit the program, it releases the video capture object and closes the display window. This ensures that system resources are properly freed up, and the program concludes its execution smoothly. In essence, this project provides real-time video processing, offering users the ability to experiment with various image filters and capture frames of interest from a live camera feed. It serves as a valuable tool for both learning and practical applications in the field of computer vision and image processing.

2) Any required images or text along with a short description of the meaning of each image or diagram.

Task 1: Read an image from a file and display it

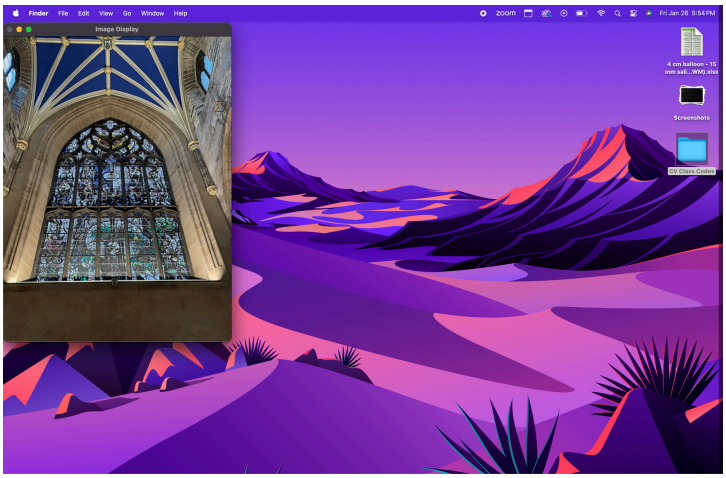

Image 0: Read an image

CS5330 - pattern Recognition and Computer Vision - Project 1
Authored by - Aadhi Aadhavan Balasubramanian & Harishraj Udaya Bhaskar

Task 3: Displaying grayscale live video

Image 1: Original Video

Image 2: grayscale filter Video

Task 4: Display alternative greyscale live video

Image 3: Built-in grayscale filter Video

Image 4: custom grayscale filter video

To achieve the custom grayscale we are subtracting the red channel value from 255 to calculate the grayscale value. This results in an inverted grayscale image where brighter regions in the original image become darker in the grayscale version, and vice versa. It's a custom transformation where the grayscale value is calculated based on the red channel of each pixel.

Task 5: Implement a Sepia tone filter

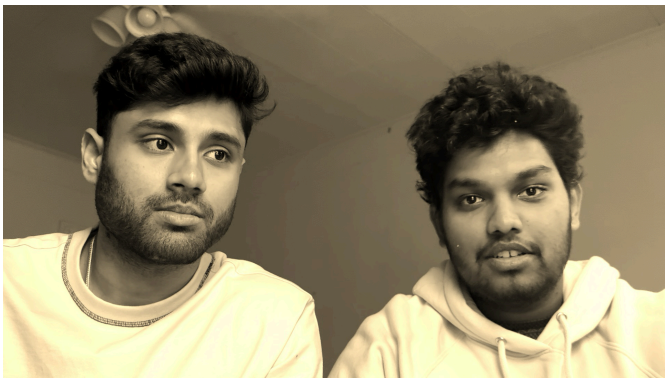

Image 5: Sepia filter applied video

The sepiaFilter function ensures the use of the original RGB values in the computation by directly accessing the RGB components of each pixel from the input image src using the color variable. It then applies the Sepia tone transformation with specific coefficients for the red, green, and blue channels. These coefficients are multiplied by the corresponding original RGB values to determine the new values for each channel.

Task 6: Implement a 5x5 blur filter

Image 6: Original video

Image 7: Blurred video

The blur effect is visible in the text if zoomed in. The effect is not very prominent due to the weights in the filter.
Timing report:

```
(base) harisha@Harishrajs-Laptop build % ./MyProject '/Users/harisha/Desktop/assignement 1/openCV setup /cathedr
al.jpeg'
Time per image (1): 0.2773 seconds
Time per image (2): 0.1159 seconds
Terminating
```

Image 8: Timing report for blurring the cathedral image

The image (1) is blurred using a Naive blurring filter
The image(2) is blurring using a faster blurring filter

CS5330 - pattern Recognition and Computer Vision - Project 1
Authored by - Aadhi Aadhavan Balasubramanian & Harishraj Udaya Bhaskar

Task 7 : Implement a 3x3 Sobel X and 3x3 Sobel Y filter as separable 1x3 filters

Image 8 : Sobel X filter video

Image 9 : Sobel Y filter video

Task 8: Implement a function that generates a gradient magnitude image from the X and Y sobel images

Image 10: Original video

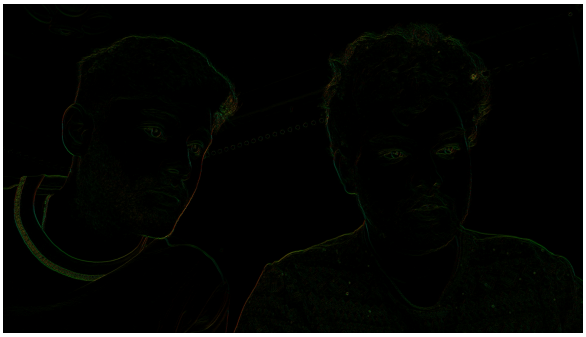

Image 11: Gradient filter video

Task 9 :Implement a function that blurs and quantized a color image

Image 12: Original video

Image 13: Quantized video

CS5330 - pattern Recognition and Computer Vision - Project 1
Authored by - Aadhi Aadhavan Balasubramanian & Harishraj Udaya Bhaskar

Task 10 : Detect faces in an image

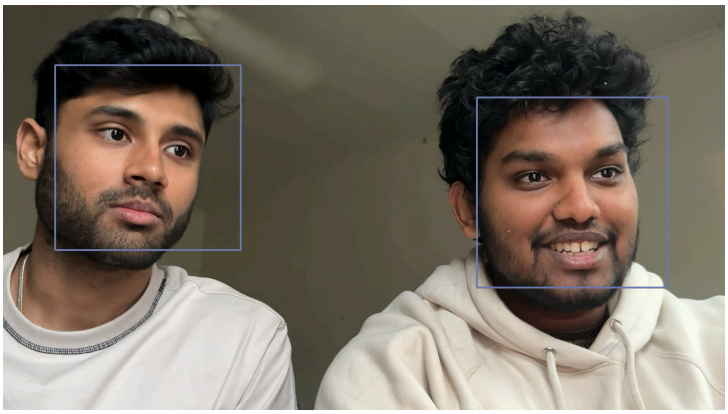

Image 14: Face detection

Task 11 : Implement three more effects on your video

a) Highlighting the strong color to remain and set everything else to grayscale.

Image 15 : Original video

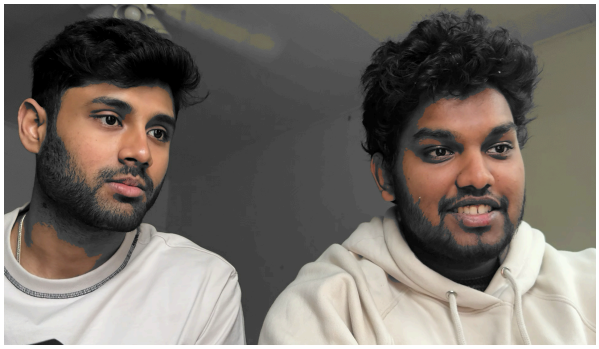

Image: 16 Dominant color highlighted

b) Blur the image outside of found faces.

Image 17: Original video

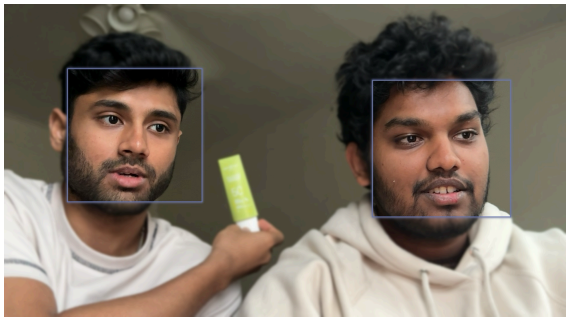

Image 18: Blurred video

c) Change around the color palette or make the image a negative of itself

Image 19: Original video

Image 20 : Negative video

Extensions:

1. **Facial privacy guard:**

   This filter identifies the faces in a video stream and selectively blurs the faces of the users. This function takes an input image, a vector of face rectangles, and selectively applies a Gaussian blur to the regions inside the detected faces. The extent of blur could be adjusted.

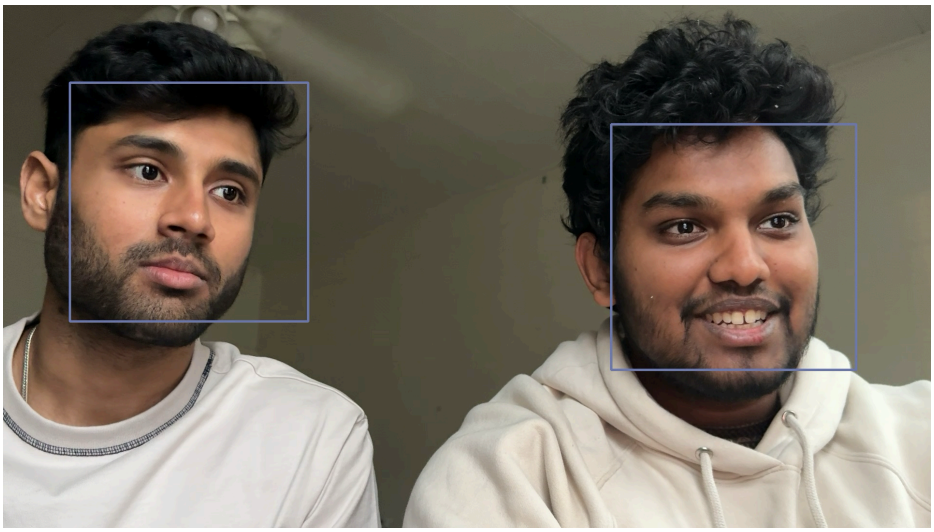

Image 20: *Original video without face blur / facial privacy*

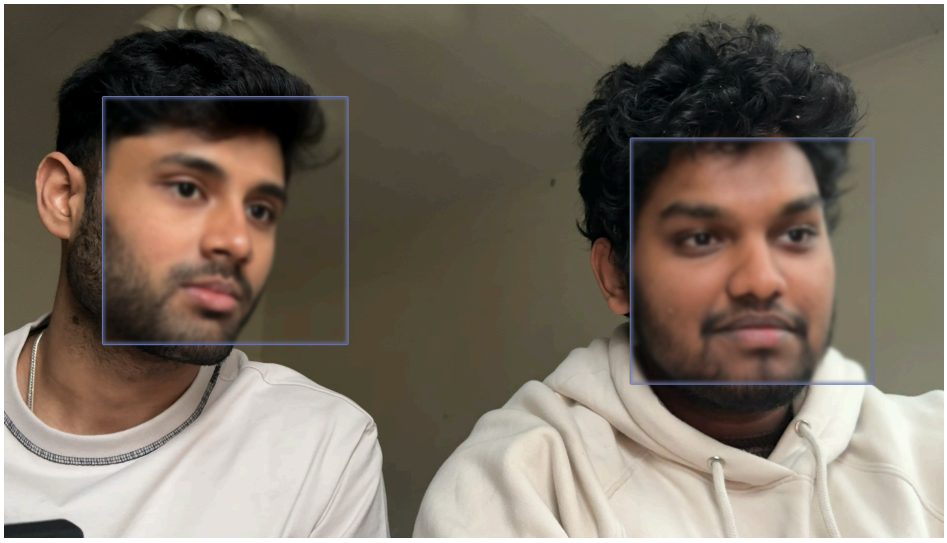

Image 21: *Video with facial privacy.*

2. **Thermal Palette:**

This filter applies a thermal palette transformation to the input image/video.

This function takes an input color image (BGR format) and converts it to a grayscale image. Then, it applies a thermal colormap to provide the appearance of temperature variations, ranging from cool to warm. The result is stored in the provided destination matrix in RGB format.

Image 22: *Video with the thermal palette on.*

3. Vertical Mirroring effect

Applies vertical mirroring to an input image/video.

This function takes an input image and produces a vertically mirrored version of it by flipping along the vertical axis. The resulting image is stored in the provided destination matrix.

Image 23: *Video with vertical mirroring on*

4. Extension 4:

Applies an embossing effect to the input image/video using the Sobel filter.

This function takes a color image in BGR format as input and applies an embossing effect by convolving the image with Sobel filters in the X and Y directions. The resulting embossing effect is stored in the output matrix.

Image 24: *Video with Embossing effect on*

5. **Extention 5:**

   Applies sparkle effect to an input image/video stream.

   This function takes an input image in BGR format, converts it to greyscale, performs Canny edge detection, and adds sparkle effects to the image where strong edges are detected. The output image is stored in the provided destination matrix.

Image 25: *Video with Sparkle effect on*

6. **Extension 6:**

   Applies a black-outside-faces effect to an input image.

   This function takes an input image (`src`) and a vector of rectangles representing detected faces (`faces`). It creates a binary mask where the face regions are set to white (255) and the rest is black (0). The original face regions are copied to the output image (`dst`), while the regions outside the faces are set to black.

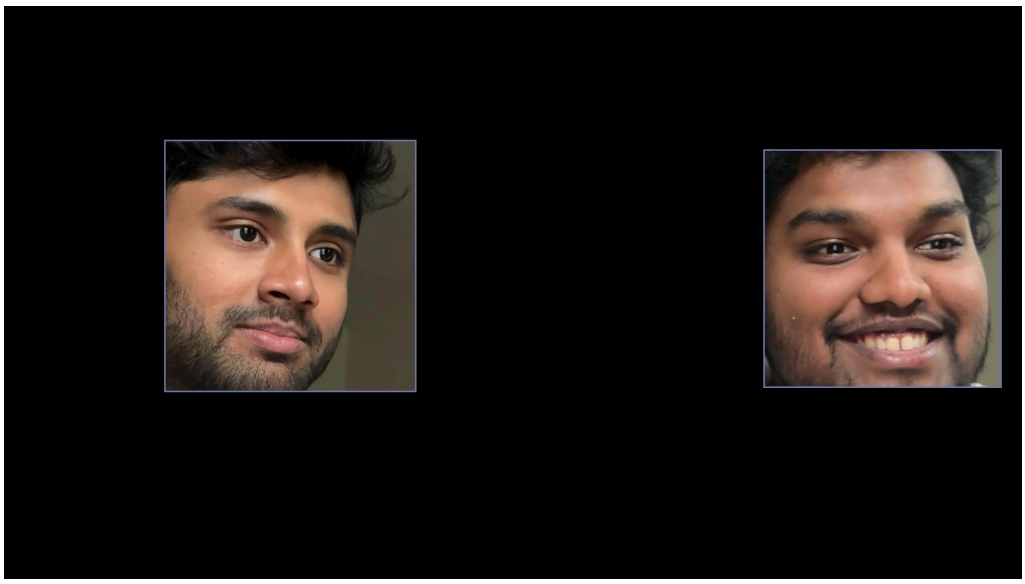

Image 26: *Video with* black-outside-faces effect *on*

CS5330 - pattern Recognition and Computer Vision - Project 1
Authored by - Aadhi Aadhavan Balasubramanian & Harishraj Udaya Bhaskar

4) A short reflection of what you learned.

In this C++ program, we learned how to work with OpenCV to capture video from a camera and apply various real-time image processing filters. We discovered how to create a keystroke interface that allows us to toggle between different filters using keyboard shortcuts. We also learned about the implementation of these filters which include grayscale, custom grayscale, sepia, blur, Sobel edge detection, gradient magnitude, color quantization, face detection, color channel retention, selective blurring inside and outside faces, thermal palette, vertical mirroring, negative image, embossing effect, and sparkles.

We implemented these filters in a separate file, which helps keep the code organized and modular. We gained insights into handling user input efficiently and saving images when needed.Overall, this project improved our understanding of computer vision and image processing using OpenCV, as well as how to create a responsive and interactive program in C++.

5) Acknowledgement of any materials or people you consulted for the assignment.

The following resources were used for the completion of this project:

- https://youtu.be/KhGnYWplLVo?si=fVH3yD233Y3K2_gi
- https://youtu.be/Ozc3zWJ_NhQ?si=6WQCXqr2diwUi-Hz
- https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html
- https://stackoverflow.com/
- Live classes and recording by Professor Bruce maxwell CS5330,

We would like to express our sincere gratitude to Prof. Bruce Maxwell for equipping us with the necessary knowledge and skills to excel in this course. Thank you for your guidance and support throughout the learning process.