Project Report

Video-special effects

Pattern Recognition and Computer Vision

CS 5330

Spring 2024

Submitted by.
Saikiran Juttu & Suriya Kasiyalan Siva

In the project, the initial tasks focus on foundational image and video processing using OpenCV in C++. The code is organized into separate files to facilitate modularity and clarity. The imgDisplay.cpp file handles image display, while vidDisplay.cpp extends this functionality to live video, introducing key commands for user interaction.

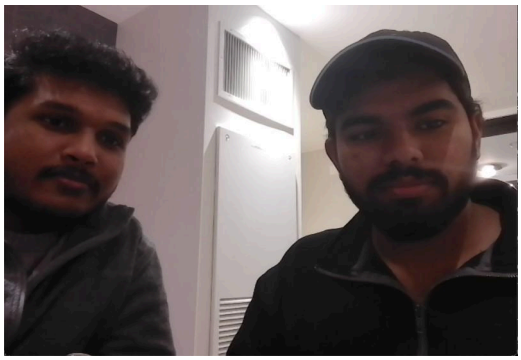
The subsequent tasks delve into more advanced image processing techniques. Greyscale modes are added to the live video display, with a custom grayscale filter providing a unique transformation. A sepia tone filter is implemented, offering an antique camera effect. The project introduces a 5x5 blur filter, initially implemented naively and later optimized for speed. Sobel X and Sobel Y filters are added, with their outputs used to calculate the gradient magnitude.

Furthermore, the project involves the development of a function for blurring and quantizing color images, allowing users to control the number of quantization levels. The face detection feature is integrated, enabling the program to locate faces in the video stream using the code files provided.

In the final stage, the following effects were incorporated on the video stream.

- Pick a strong color to remain and set everything else to grayscale.
- Allow the user to adjust brightness or contrast.
- Blur the image outside of found faces.

1. Greyscale live video



Original Image

cvtColor version of the greyscale image

2. Alternative greyscale live video

In order to create a custom grayscale function we subtracted the red channel from 255 and copied the value to all three color channels.

In contrast, the cv::cvtColor function provides a more efficient and flexible way to perform color space conversions, including grayscale conversion

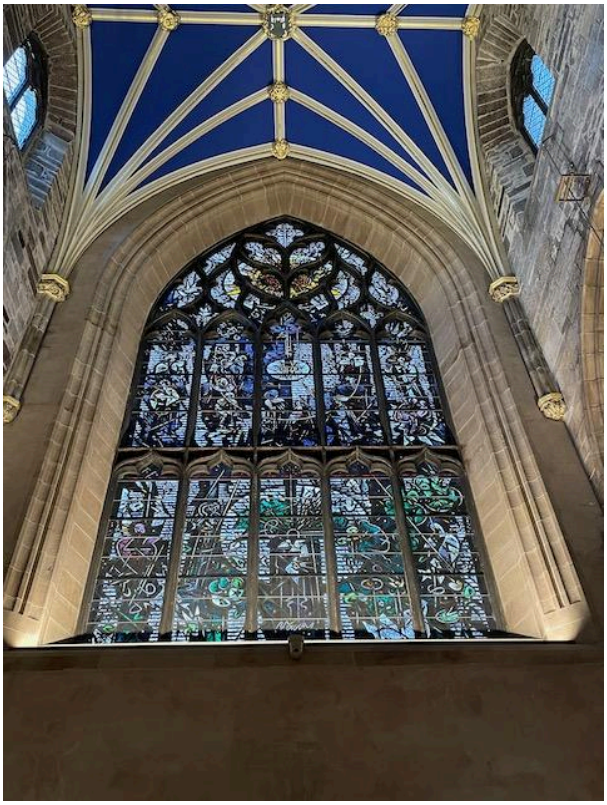
customized grayscale image

3. Sepia Tone filter

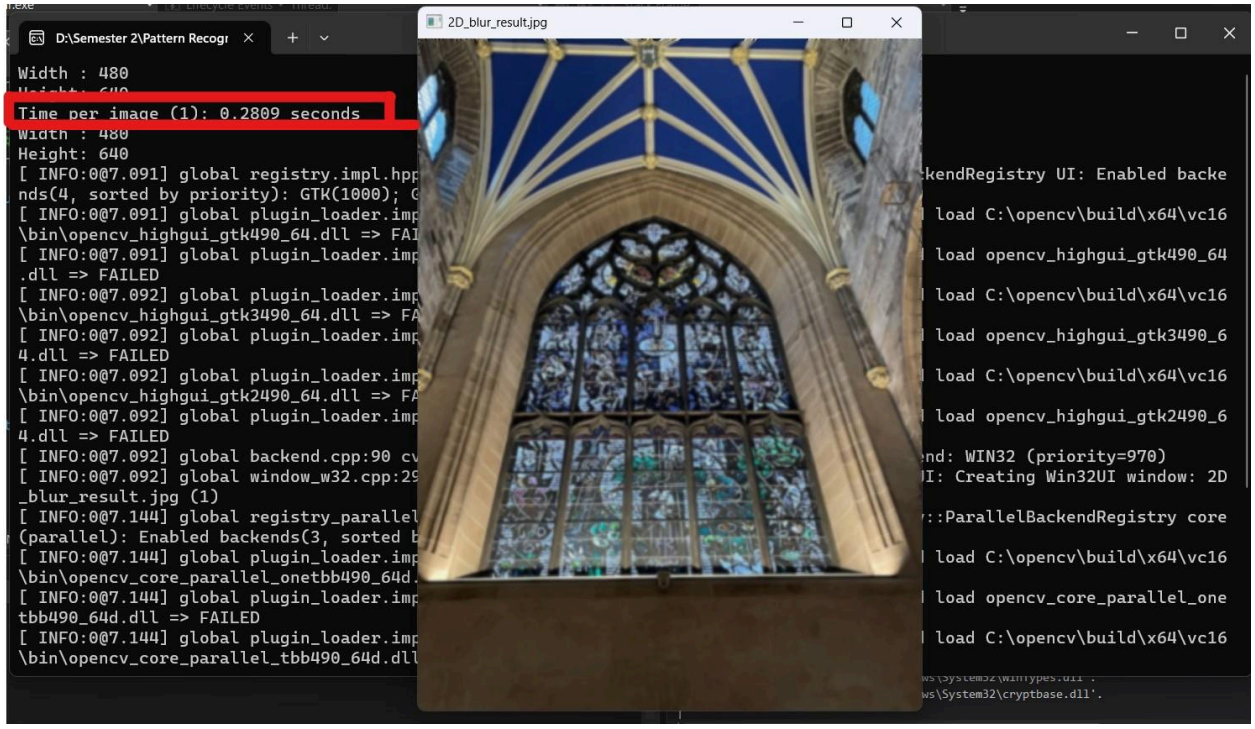
In the calculations, red, green, and blue are the original color values. The computed newRed, newGreen, and newBlue values are then used to update the pixel's color channels. This ensures that each channel is modified based on the original values and not the newly computed values. The use of the cv::saturate_cast<uchar> ensures that the resulting values are saturated to the valid range [0, 255].

Sepia tone filter image

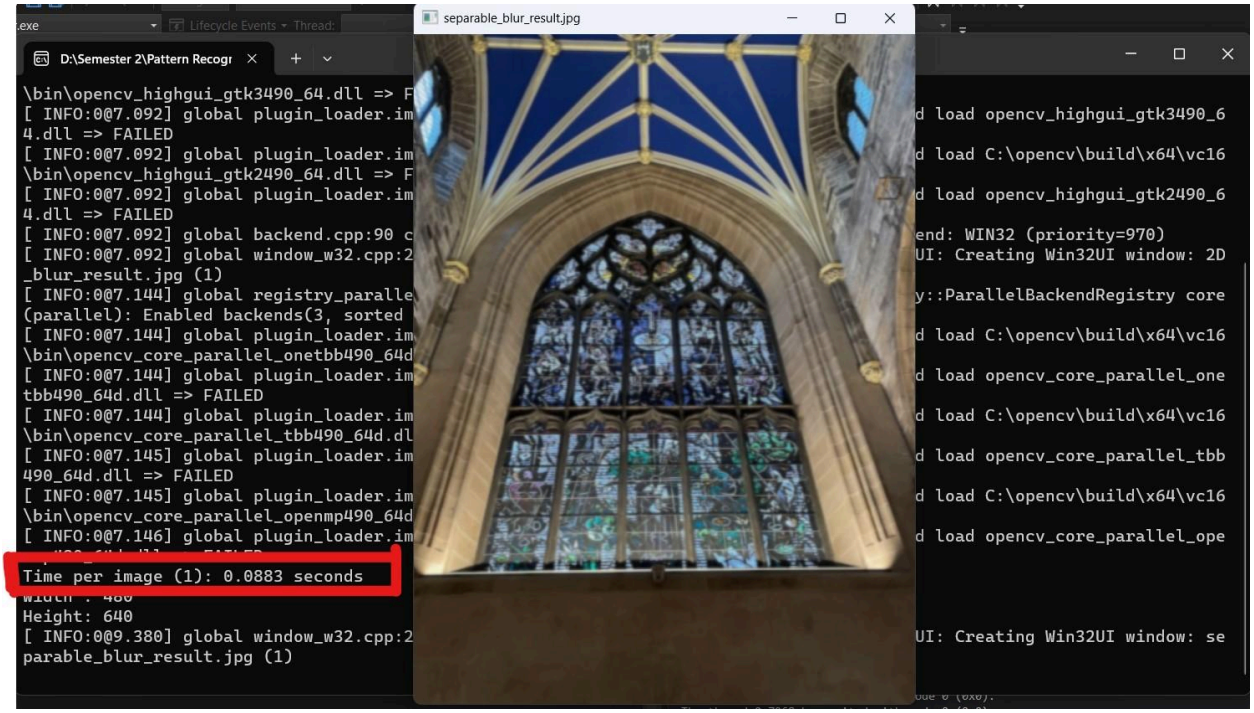
4. Blur filter



Original image



Blur using 2D Guassian filter



Blur using separable 1D filters.

As we can notice that there is a drastic change in Implementation time when we use 1D separable filters instead of 2D filters. Implementation time is reduced by half.

5.Blur and quantize a color image

Original Image

Blur quantize Image

6. Face detection filter

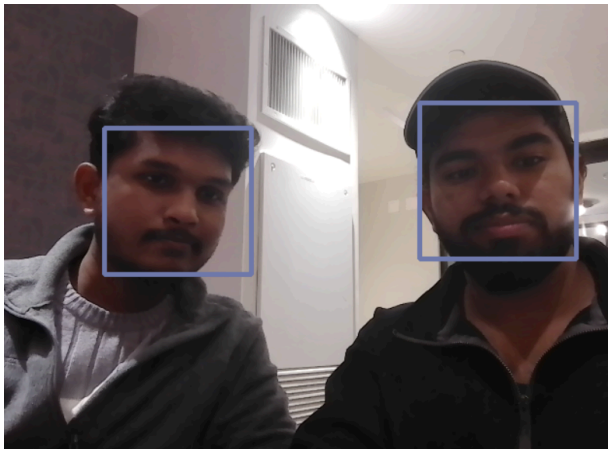
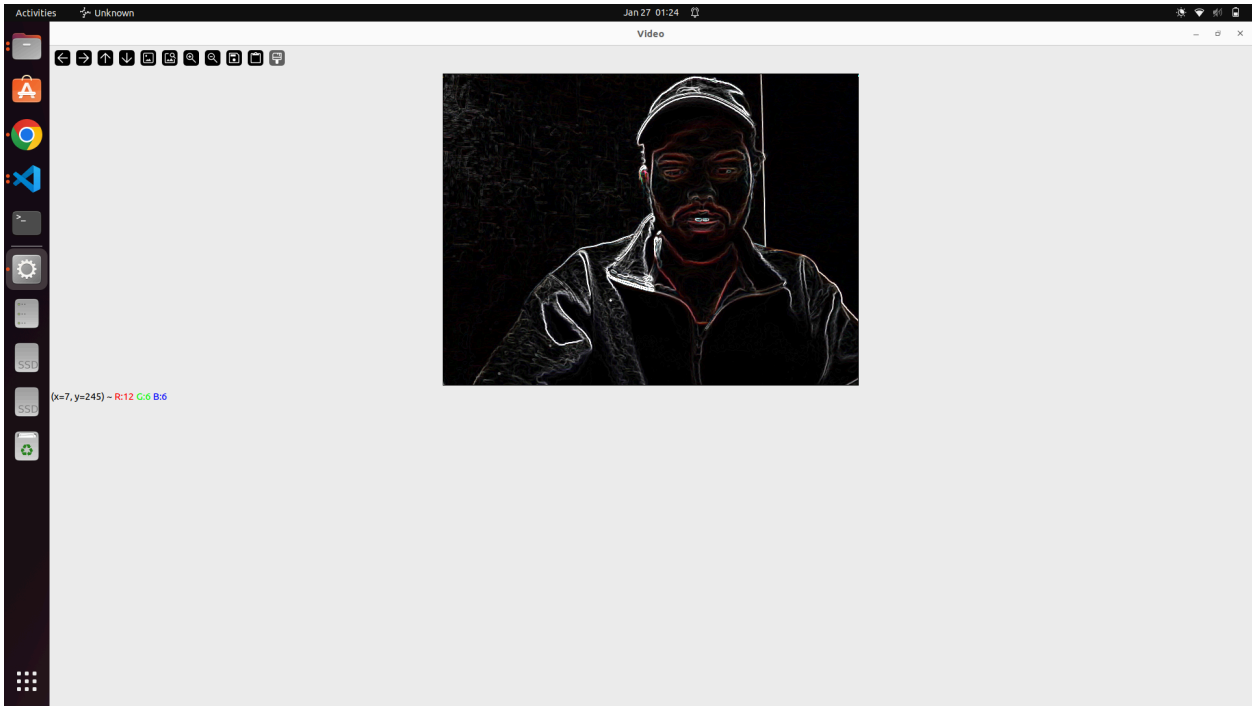


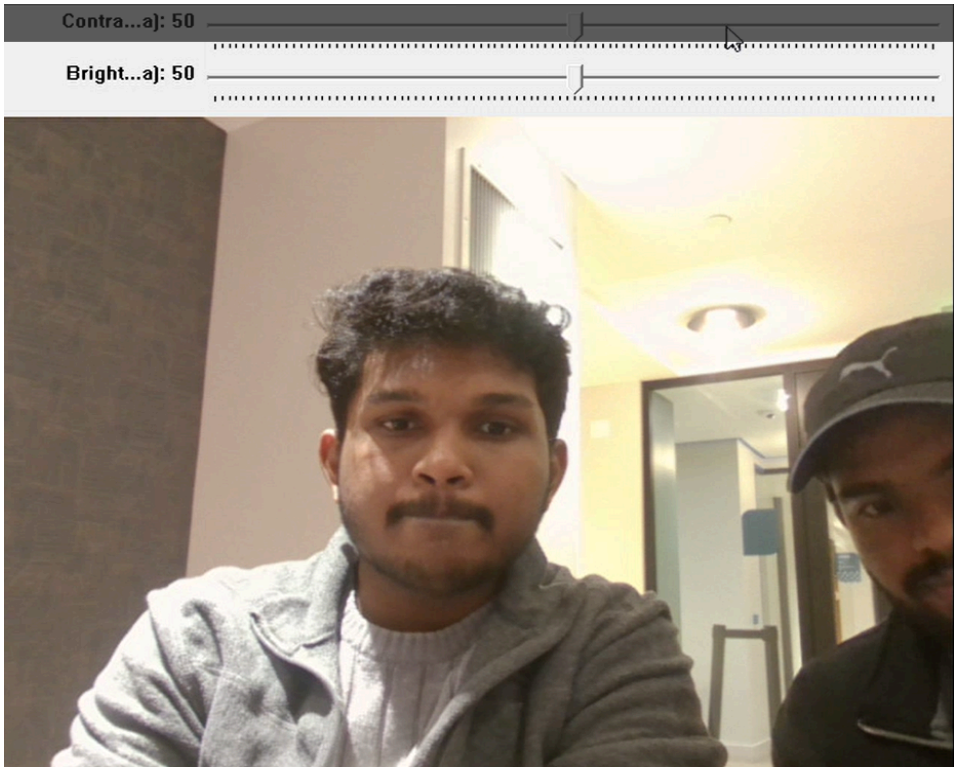
Image showing the detected faces

7.Sobel X and Y images

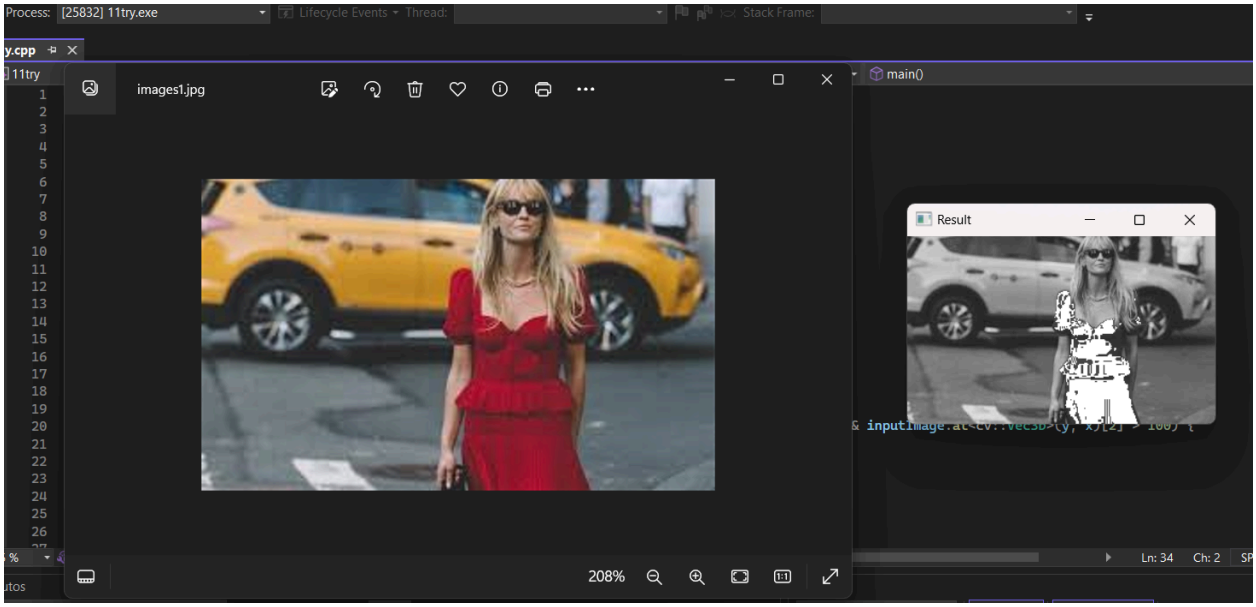


Sobel gradient magnitude

8. **Adjust brightness or contrast.**



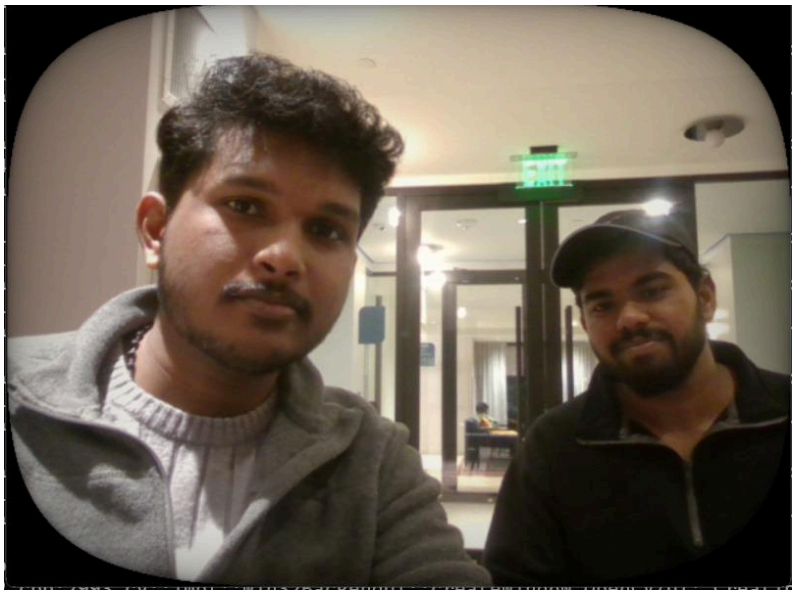
9. Strong color to remain and set everything else to greyscale



Extensions

1. **Vignetting to sepia tone filter.**

Combining vignetting with a sepia tone filter creates a nostalgic and vintage aesthetic. Vignetting darkens the image edges, drawing focus to the center, while sepia toning adds a warm, brownish tint reminiscent of old photographs. This combination enhances the overall antique feel of the image, giving it a timeless and classic look.



Short reflection

Throughout this project, We have acquired a comprehensive understanding of computer vision concepts and practical applications through the lens of OpenCV in C++.

Key takeaways include handling image and video data, implementing diverse filters, and integrating interactive features such as keypress detection. The project allowed me to explore fundamental concepts like grayscale conversion, blur filters, and advanced techniques including Sobel operators for edge detection.

Working on the project honed our ability to navigate OpenCV documentation and other resources for effective problem-solving.The project's modular structure emphasized the importance of code organization and comments, facilitating better maintainability and readability.

Overall, this project significantly enhanced our proficiency in computer vision, bolstering confidence in applying image processing techniques to solve real-world problems.

Acknowledgement

1. OpenCV Documentation
2. CPP Reference
3. OpenCV Tutorials
4. Stack Overflow Community
5. Installation