CS 5330 Project One Report

Name: Hang Zhao

Email:zhao.hang1@northeastern.edu

Team member: Hang Zhao(For this project just myself. I will try to find a teammate when I submit a project next time.)

Project Description:
In this project, I start by setting up OpenCV and creating a program to display a static image. Next, we extend it to capture live video and apply real-time effects like grayscale, sepia, Gaussian blur, and Sobel filters. We also add features like face detection and depth-based effects using the Depth Anything V2 network. To wrap it up, we create three custom filters: a frosted glass effect, a time delay effect, and a color enhancement filter, making the video stream more dynamic and interactive.

Images and description
Task 3:

original image: This image represents the raw video frame captured directly from the live camera feed in its original color format.

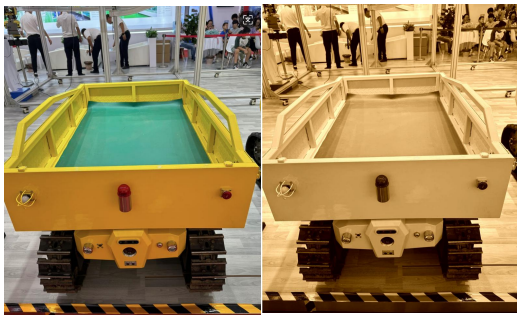
Grayscale Image (Using cv::cvtColor):
This image shows the same video frame after being converted to grayscale using OpenCV's cvtColor function. In this transformation, the color information is removed, and pixel intensity is computed using weighted contributions of the original RGB channels:
Gray = 0.299*Red + 0.587*Green + 0.114*Blue.

Task 4:

In the custom grayscale image, I chose the following channel weight ratios:
Red (R): set to 0 to completely ignore the contribution of the red channel.
Green (G): set to 30% to emphasize the influence of the green channel.
Blue (B): set to 40% to further enhance the contribution of blue.
Total is negative 8%: This overall adjustment reduces the brightness of the grayscale image, making it darker and more detailed.

Default grayscale image:
Generated by the standard formula Gray = 0.299 * R + 0.587 * G + 0.114 * B.
The default grayscale image evenly considers the weights of the red, green, and blue channels, and can truly reflect the brightness ratio of different colors in the image. Suitable for the overall conversion of ordinary scenes, with a natural transition between light and dark.
Custom grayscale image:
The red channel is completely ignored, so the brightness of red-dominated areas (such as skin, flames, etc.) is significantly reduced. Green and blue are enhanced to highlight the layering of grass, sky, or water in natural scenes. Since the total amount is negative 8%, the overall brightness is reduced, making the picture appear more stable and profound, adding a unique cold tone atmosphere.

Task 5:

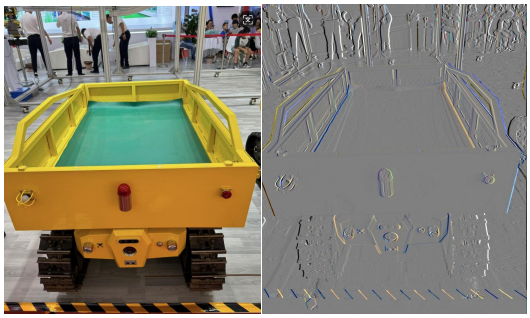


The Sepia filter calculates the new RGB value for each pixel using the following matrix formula:
New Blue = 0.272*R + 0.534*G + 0.131*B
New Green = 0.349*R + 0.686*G + 0.168*B
New Red = 0.393*R + 0.769*G + 0.189*B
These weights accentuate warm tones (reds and browns), simulating the effect of vintage photography. The Sepia filter always uses the original pixel's R, G, B values, each time it calculates new R, G, B values, preventing the modified values, from affecting subsequent calculations.

Task 6:

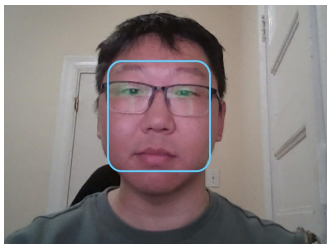
```
Time per image (1): 0.2403 seconds
Time per image (2): 0.1371 seconds
```

Task 8:

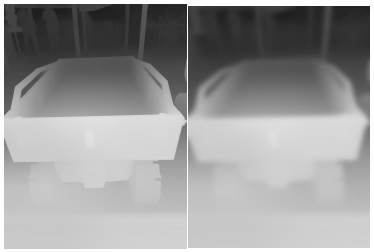


Task 9:

Task 10:

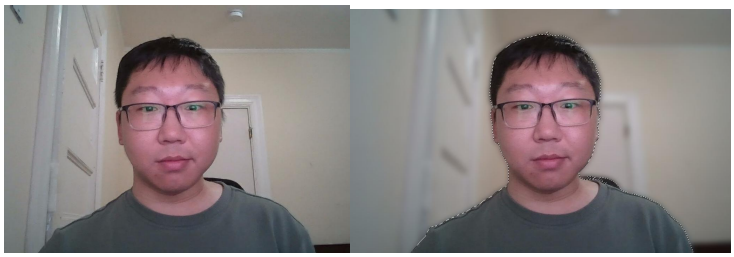


Task 11:

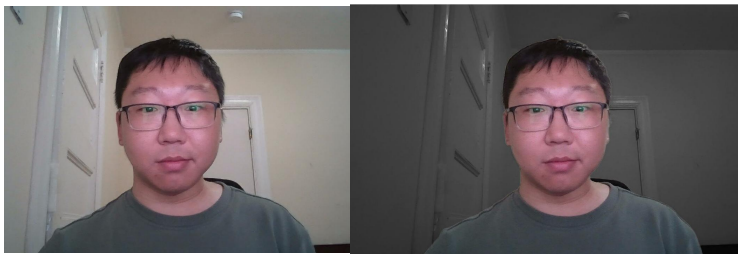


Task 12:

Blur the area outside the face



Face is colored, other parts are grayscale



Red Enhancement

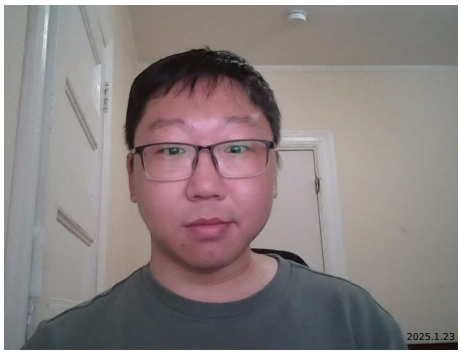
Extensions:

Implement a function that allows users to save static images with special effects applied to local storage and automatically add special effects watermarks to the images.

filter.cpp: addWatermark function

vidDisplay.cpp: add logic when key == w

After the user applies special effects to the video stream, press the w key to save the current frame. Saved images are automatically watermarked.



Reflection:

Through this project, I learned how powerful OpenCV is for image and video processing. Starting with simple tasks like displaying an image, I gradually built a program that can apply real-time effects, detect faces, and even use depth estimation. Creating custom filters like frosted glass and time delay was both challenging and fun, letting me experiment with creative ideas. This project gave me hands-on experience with computer vision concepts and showed me how to implement them in an interactive way. Overall, it made me more confident in using OpenCV and excited to explore more advanced topics.

Acknowledgement:

I would like to thank our teaching assistants and classmates for their valuable support and guidance throughout this project. Their help in troubleshooting technical issues and providing feedback was essential in completing our work.

I also want to express our gratitude to our professor for providing clear explanations and insightful examples that deepened our understanding of computer vision. The resources and tutorials shared during the course were incredibly helpful in learning OpenCV and applying it effectively to this project.