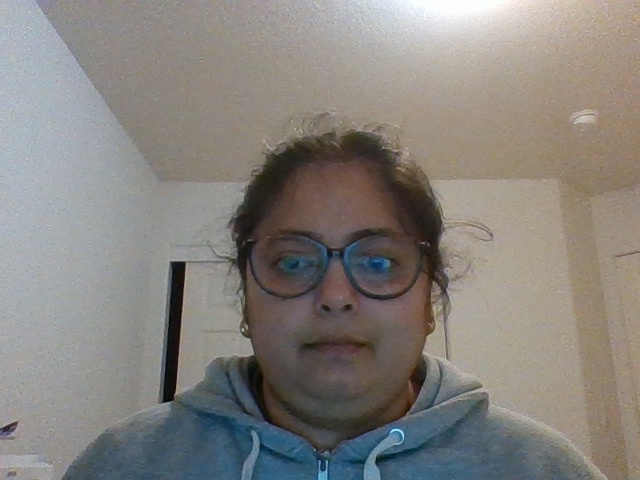
**Project 1: Video-special effects**

**1. Project Overview:**

This project explores the field of image processing on video streams by utilizing the C++ OpenCV package. The project's key component is its ability to apply a wide range of image processing techniques on a live video input. These techniques include fundamental conversions like grayscale and more sophisticated ones like face detection and custom filter effects. Through pixel-by-pixel manipulation of image data, custom filter implementation, and integration of face identification methods, the project reflects the complexity of OpenCV. This thorough examination of both basic and sophisticated facets of image processing highlights OpenCV's dependability and adaptability for real-time image processing and analysis.

**2. Visual Demonstration and Descriptions**

**Original Image:**

****

**Figure 1: Original Image**

**Grayscale Image**

****

**Figure 2: Grayscale Image**

**Grayscale Image**: The grayscale version is generated using the **cvtColor** function, which converts the color space from BGR to grayscale. This standard method is widely used for grayscale conversion.

Note: Refer original Image (Figure 1: Original Image) at Page 1

**Custom Grayscale Image**

****

**Figure 3: Custom Grayscale Image**

* **Custom Grayscale Image**: This image is generated by subtracting the green channel from 255 and applying this value to all three-color channels (R, G, B). This method creates a unique grayscale effect where the green components significantly influence the outcome, differing from the standard grayscale conversion which weighs color channels based on human perception.

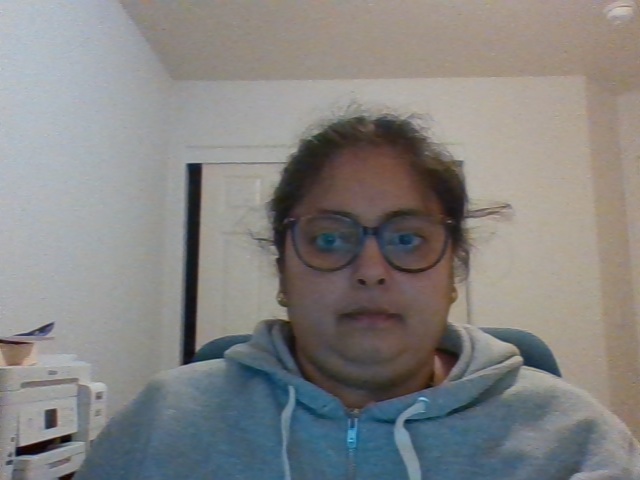
**Sepia Tone Filter**



**Figure 4: Sepia Tone Image**

* **Sepia-Toned Image**: The sepia filter transforms the image to mimic the brownish tone characteristic of old photographs. The filter is applied by adjusting each pixel's color values using specific coefficients that simulate the sepia effect.

**Blurred Image**

****

**Figure 5: Blur Image**

* **Blurred Image**: This blurring is achieved through a custom 5x5 blurring method using separable matrix. The time taken to blur an image using this custom method was approximately 37 milliseconds. This demonstrates a significant efficiency gain compared to the time taken by the Gaussian blur filter, which was approximately 513 milliseconds. This substantial difference in processing time underscores the efficiency of the separable matrix approach in image blurring operations.



**Figure 6: Timing Information**

Note: Refer original Image (Figure 1: Original Image) at Page 1

**Gradient Magnitude Image**



**Figure 7: Gradient Magnitude Image**

* **Gradient Magnitude Image**: This image is created by applying the Sobel operator in both X and Y directions to emphasize edges and gradients in the image. The resulting image showcases the intensity of gradients.

Note: Refer original Image (Figure 1: Original Image) at Page 1

**Blurred - Quantized Image**



**Figure 8: Blurred - Quantized Image**

* **Blurred - Quantized Image**: After applying a blur effect, the image is quantized into discrete color levels, giving it a posterized look.

Note: Refer original Image (Figure 1: Original Image) at Page 1

**Face Detection Image**



**Figure 9: Face Detection**

* **Face Detection in Video Stream**: This image demonstrates the functionality of face detection in a video stream. It shows the original frame with detected face highlighted by rectangles.

**Embossing Effect**

****

**Figure 10: Embossing Effect**

* **Embossing Effect**: The 't' keystroke activates an embossing effect, achieved by using Sobel filters in X and Y directions. The algorithm calculates the dot product of Sobel outputs with a direction vector (e.g., (0.7071, 0.7071)), creating a raised or carved appearance on surfaces and edges in the video stream.

**Pick a strong color to remain and set everything else to greyscale**



**Figure 11: Original Image**



**Figure 12: Selective Color Retention**

* **Selective Color Retention**: Activated by the 'a' keystroke, this feature selects a strong color (in this case, a color with the RGB values (18, 17, 74)) to keep, turning all other parts of the image to grayscale. This selective color technique can be used to highlight specific objects or areas within the video stream, making them stand out against the de-saturated background.

**Cartoonization Effect**

****

**Figure 13: Cartoonization Effect**

* **Cartoonization Effect**: The 'j' key enables the cartoonization of the video stream. It simplifies the color palette and emphasizes edges to create a cartoon-like effect. This is achieved through a combination of blurring, edge detection, and color quantization, which together create a stylized version of the original video feed.

**Blur the image outside of found faces Effect.**

****

**Figure 14: Blurred Background with Sharp Faces**

* **Blurred Background with Sharp Faces**: This feature blurs everything in the video stream except for detected human faces. It is activated by the 'r' keystroke. When a face is detected using Haar cascades, the rest of the image is blurred using a Gaussian filter, while the region containing the face remains sharp and clear. This effect focuses the viewer's attention on the faces while providing privacy for the surroundings.

**Allow the user to adjust brightness or contrast.**

****

**Figure 15: Brightness Adjusted Image**



**Figure 16: Contrast Adjusted Image**

* **Brightness/Contrast Adjusted Image**: The 'd' and 'e' keystrokes, combined with mouse scroll actions, allow real-time adjustments to brightness and contrast of the video stream. Brightness is modified by altering the lightness or darkness across all pixels, while contrast adjustments amplify or reduce the difference between the lighter and darker elements of the image.

**Make the face colorful, while the rest of the image is greyscale Effect**



**Figure 17: Colorful Face on Grayscale**

* **Colorful Face on Grayscale**: By pressing the 'c' key, the algorithm detects faces and retains their color, while the rest of the image is converted to grayscale. This effect draws attention to the human subjects within the video stream by providing a stark contrast between the colorful faces and the monochromatic background.

**Swap Faces**

**A person holding a statue

Description automatically generated**

**Figure 18: Face Detection**



**Figure 19: Swap Faces Mode**

* **Swap Faces Mode**: Activated by the 'k' keystroke, this feature detects faces in the video stream and swaps them. The algorithm identifies facial features and interchanges them between different faces in the frame. This creates a humorous and sometimes surreal effect, ideal for playful applications or artistic experimentation.

**Sepia Tone with Vignette**



**Figure 20: Sepia Tone Filter**



**Figure 21: Sepia Tone with Vignette**

* **Sepia Tone with Vignette**: Triggered by the 'v' keystroke, this mode applies a sepia filter, overlayed with a vignette effect, to the video stream. The sepia filter adds a warm, brownish tone, reminiscent of vintage photographs, while the vignette effect darkens the edges of the frame, focusing attention on the center. This combination creates a nostalgic and focused visual experience.

**A Short Reflection of What I Learned**

Throughout this project, I gained a deeper understanding of various aspects of image processing, particularly in the context of OpenCV and C++. One of the key insights I gathered was the efficiency of accessing and manipulating images using pointer methods. By directly interacting with pixel data, I was able to appreciate the significant difference in time complexity compared to the **at** method, thereby grasping the importance of optimization in real-time image processing applications.

I explored a range of filtering techniques, including median filters which utilize neighboring pixel values for operations. This not only enhanced my knowledge of image filtering but also provided practical insights into how different filters impact image quality and the perception of details.

A significant part of my learning revolved around understanding color channels. I learned when to utilize three-channel color images and when a single-channel grayscale image is more appropriate. Creating matrices of different sizes and channels equipped me with the flexibility to manipulate images according to various processing requirements.

Face detection was another intriguing aspect of my learning journey. Understanding the mechanics behind facial feature recognition and how to implement these in a video stream was particularly rewarding.

Lastly, embedding new pixel data into existing images was a fascinating exercise. It allowed me to creatively alter images and understand the underlying data structures that make up digital images.

In conclusion, this project was a comprehensive and hands-on learning experience that significantly enhanced my understanding of image processing and computer vision techniques.

**Acknowledgements**

This project was enriched and made possible by a range of resources and materials that provided invaluable guidance and insight. I would like to acknowledge the following sources:

**Textbooks:**

* "Computer Vision: Algorithms and Applications, 2nd Edition" by Richard Szeliski: This textbook was instrumental in providing foundational knowledge and advanced concepts in computer vision, which were crucial for the successful completion of this project.
* "Real-Time Video Abstraction" by Holger Winnemöller, Sven C. Olsen, and Bruce Gooch: The techniques and methodologies discussed in this book were pivotal in understanding real-time image processing and video abstraction techniques.

**Websites:**

* HIPR2 (<https://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm>): Provided comprehensive information on median filters, enhancing my understanding of their applications in image processing.
* University of Auckland (<https://www.cs.auckland.ac.nz/courss/compsci373s1c/PatricesLectures/Image%20Filtering.pdf>): This resource was essential in understanding the principles and implementation of various image filtering techniques.
* Wikipedia (<https://en.wikipedia.org/wiki/Image_embossing>): Offered a clear explanation of image embossing, which was key in implementing this feature in the project.
* Grokking the GIMP (<https://tinf2.vub.ac.be/~dvermeir/manual/gimp/Grokking-the-GIMP-v1.0/node74.html>): This guide provided insights into advanced image manipulation techniques which were applied in the project.
* LearnOpenCV (<https://learnopencv.com/photoshop-filters-in-opencv/>): This website was a valuable resource for understanding how to implement various Photoshop-like filters using OpenCV.
* OpenCV Forum (<https://forum.opencv.org/t/face-detection-for-static-image-find-top-of-head-and-chin/3009>): This forum was particularly helpful in troubleshooting and refining the face detection feature of the project.

**Personal Acknowledgements:**

* I would also like to express my gratitude to my instructors who provided guidance and support throughout this project. Their feedback and suggestions were crucial in overcoming challenges and enhancing the project's outcomes.

These resources, combined with the support from my academic environment, played a vital role in the successful execution of this project. I am grateful for the wealth of knowledge and expertise shared by these sources.