CS 5330: Pattern Recognition and Computer Vision (Spring 2024)

Report

Project 1: Video-special effects

Submitted by: Haard Shah
Hrigved Suryawanshi

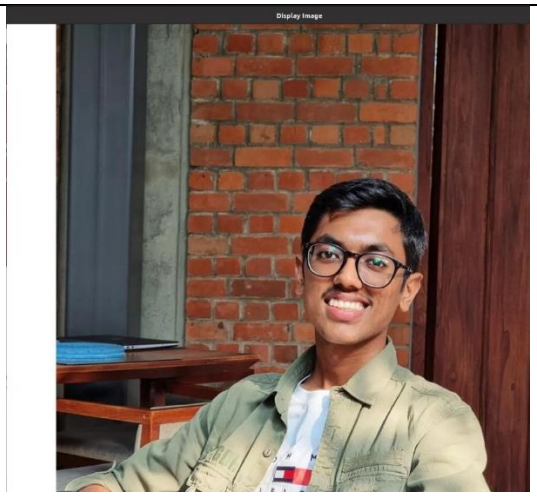
Description:

The Video-special effects project aims to get us acquainted with the C/C++ OpenCV package. The assignment progresses through various stages, starting with basic image reading and display functionalities and advancing to live video streaming from an integrated webcam. Throughout the project, participants will gain hands-on experience in the dynamics of opening, capturing, manipulating, and writing images. Additionally, the project will involve the implementation of face detection in a live video stream and the application of effects to augment the understanding of image processing concepts.

Tasks:

1. **<u>Read an image from a file and display it</u>**

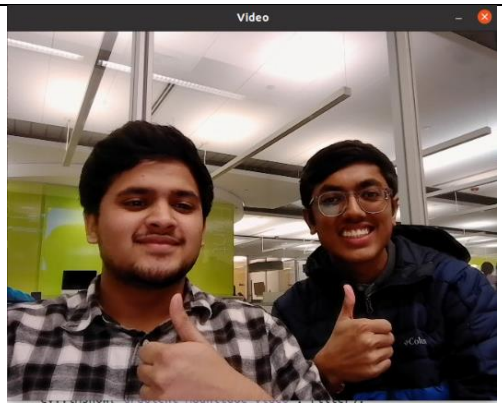
- The original image stored in the file is read and displayed.



Display Image (Fig.1)

2. **<u>Display live video</u>**

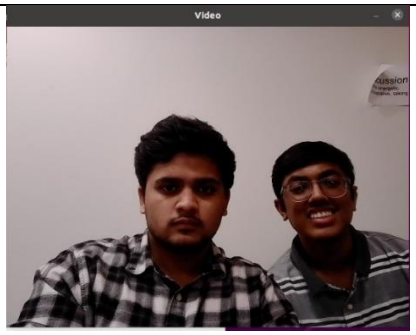
- In this, the program shows the live video by accessing the integrated camera
- Input command> ./ vidDisplay



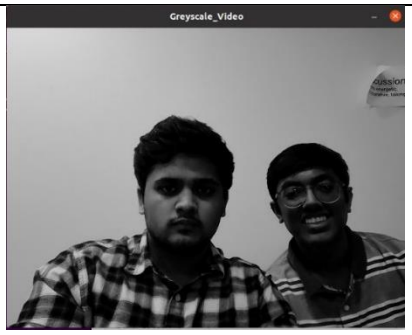
Live Video Display (Fig.2)

3. Display greyscale live video

- The original frame from the video through the integrated webcam is converted to a greyscale frame using OpenCV **"cvtColor"** function, 'COLOR_RGBA2GRAY' parameter.
- This conversion is done using the function: Y ← 0.299xR + 0.587xG + 0.114xB



Original Image (Fig.1)



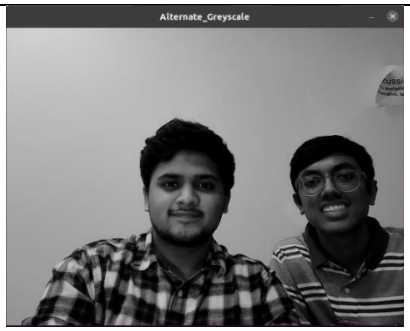
Greyscale Image (Fig.2)

Individual values are applied to the RGB color channels of the original frame from live video to get a greyscale frame. We have allocated the "g" key for this conversion.

4. Display alternative greyscale live video

- The original frame from the video through the integrated webcam is transformed to a greyscale frame by setting each pixel value to average value of the RGB channel of that particular pixel from original frame.
- Output function> (R+G+B)/3

Original Image (Fig.1)

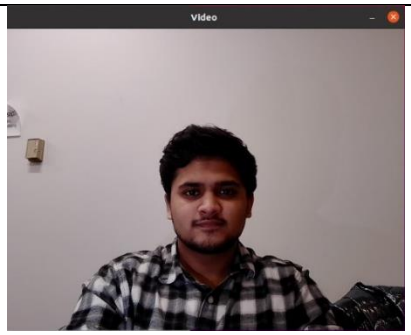


Alternate Greyscale (Fig.2)

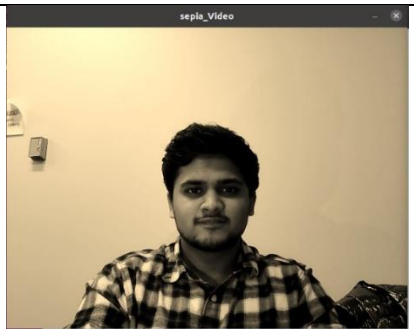
We have allocated the "h" key for this conversion.

5. Implement a Sepia tone filter

- The original frame from the video through the integrated webcam is converted to a Sepia tone filter. It makes an image look like an image clicked through an antique camera.
  Red value: 0.272*R + 0.534*G + 0.131*B
  Green value: 0.349*R + 0.686*G + 0.168*B
  Blue value: 0.393*R + 0.769*G + 0.189*B



Original Image(Fig.1)



Sepia Tone filter (Fig.2)

We have allocated the “o” key for this conversion.

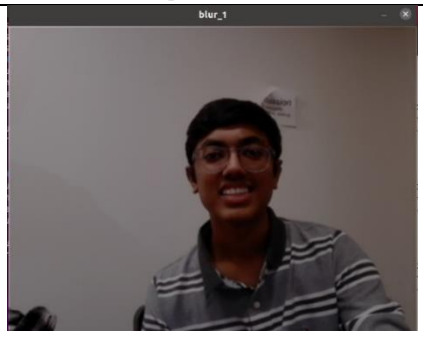
- As an extension of the sepia tone filter we have added vignetting effect (Image getting darker towards the edges) to this filter and compared that in the “Extensions” section of this report.

6. Implement a 5x5 blur filter

Implementation 1:

- For this method we have used naïve implementation of a 5x5 blur filter from scratch (“at method” to access pixels and implement filter using a single nested loop).
- It should blur each color channel separately. Use the integer approximation of a Gaussian [1 2 4 2 1; 2 4 8 4 2; 4 8 16 8 4; 2 4 8 4 2; 1 2 4 2 1].

Original Image (Fig.1)



Implementation of “at method” (Fig.2)

Implementation 2:

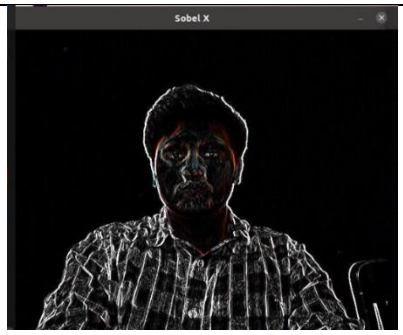
- The original frame from the live video is converted to a blurred frame by using a 5x5 Gaussian filter. We have implemented this filter as two separable 1x5 and one [1 2 4 2 1] vertical filter and another [1 2 4 2 1] horizontal filter.
- We have allocated “b” key for this conversion.

“at method” (Fig.1)

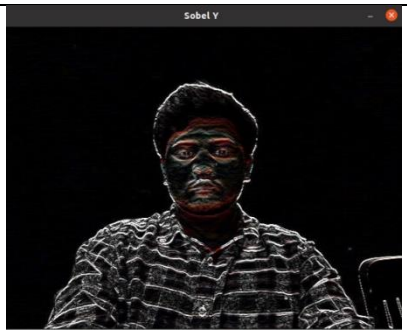
Implementation 2 (Fig.2)

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

- The original frame from the live video is converted to filtered frame using a 3x3 Sobel filter. A 3x3 Sobel X filter is implemented as two separable 1x3 filters.
- Here, as we can see in the image there is a checkered shirt where we can classify the Sobel filter in both the X and Y implementation.



Sobel X filter



Sobel Y filter

8. Implement a function that generates a gradient magnitude image from the X and Y Sobel images

- The original frame is converted to a filtered frame (Fig. 8) by a function that generates a gradient magnitude image based on Euclidean distance for magnitude.
- This conversion is done using the function: L $\sqrt{(sx)^2 + (sy)^2}$, where sx and sy are outputs from Sobel X and Sobel Y.
-

Gradient Magnitude Image (Fig.8)

- This conversion is done when the user enters the 'm' key.

9. Implement a function that blurs and quantizes a color image

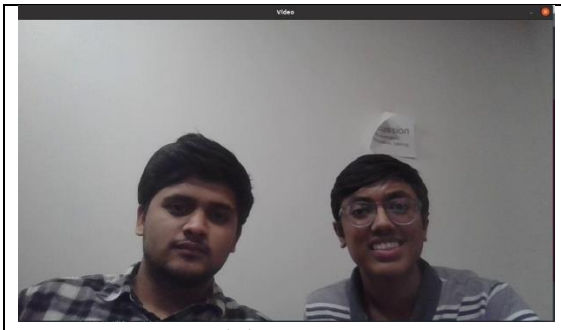
- The original frame from the video is first subjected to a blurring process, and then the image is quantized into a fixed number of levels as specified by a parameter. This quantization is achieved by using the formula b = 255/levels. , where the color channel value x and execute xt = x / b, followed by the statement xf = xt * b. After executing this for each pixel and each color channel, the image will have only levels**3 possible color values.

Original Image (Fig.1)

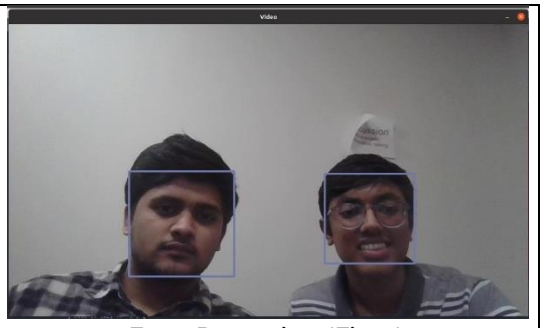
Quantised Image (Fig.2)

10. Detect faces in an image

- In this we implemented a standalone program that locates faces and puts boxes around them.



Original Image



Face Detection (Fig.2)

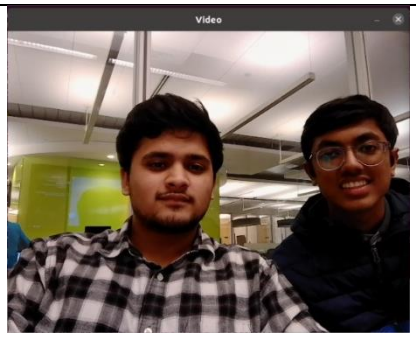
Here, we have allocated the “f” key for this execution.

11. Implement three more effects on your video

The three effects would be:

Canny Edge

- The Canny edge detection technique is a widely used method in image processing, employing a multi-stage algorithm to identify significant edges in an image by detecting areas of rapid intensity change. This process helps accentuate important structural features, making edges more prominent in the processed image. The algorithm's strict definition and optimality in meeting edge detection criteria have contributed to its popularity in various computer vision systems

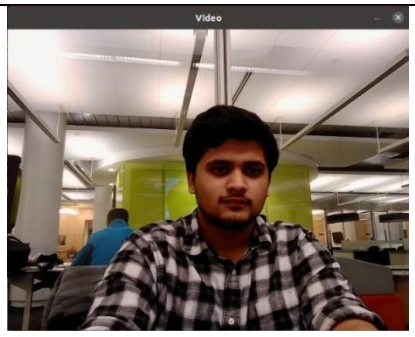


Original Image

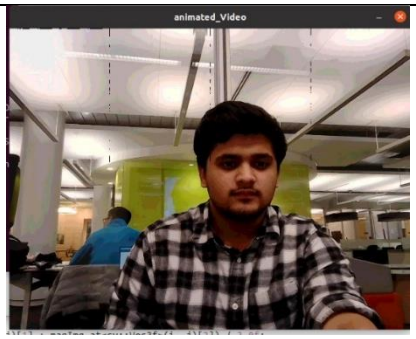
Canny Edge Filter (Fig.9)

Animated Effect

- The animated effect is a transformative process that stylizes and simplifies images, reminiscent of cartoons. This effect is achieved through a combination of edge detection and color simplification techniques, enhancing prominent features, accentuating edges, and reducing detail to create visually appealing and artistic representations



Original Image (Fig.1)

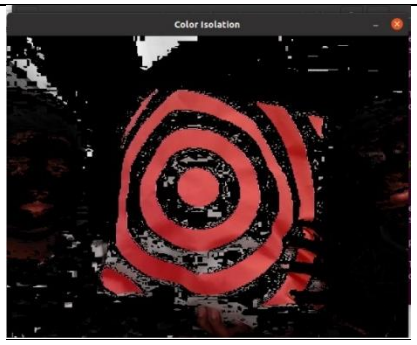


Animated Effect (Fig. 2)

Color Isolation

- Color isolation is a technique that involves highlighting specific colors in an image while desaturating the rest. By selecting and preserving certain hues, this method draws attention to particular elements, creating striking visual effects and is often used for creative expression or to emphasize specific objects in photographs

Original Image



Color Isolated Image

Here, for this project we have assigned different "keys" to execute the tasks individually.

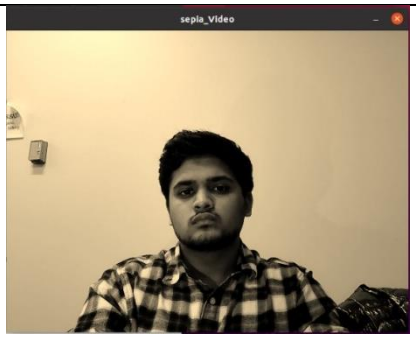
Details of Extensions:

As a part of the extension, we have made implemented three filters namely:

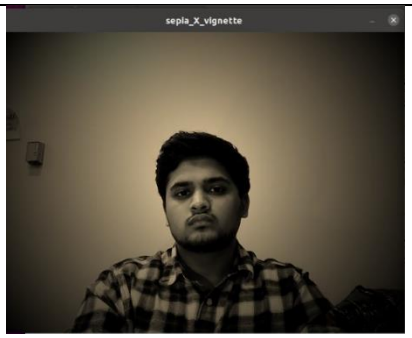
1. Vignette filter
2. High-pass filter
3. Color Inversion filter

- The Vignette filter
  - Gradient mask (center clear, edges opaque)

- Multiply sepia image with mask
- Tweak vignette opacity & explore blending modes



Sepia tone filter (Fig.1)



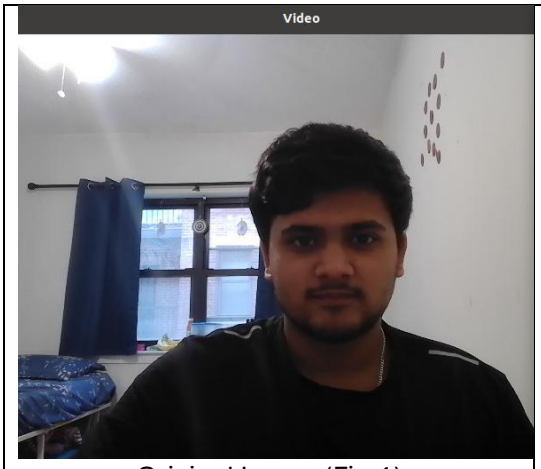
Vignetting filter (Fig.2)

We have allocated the “v” key for this conversion.

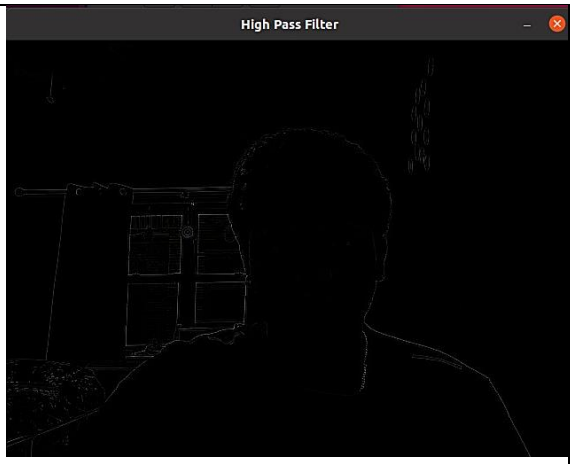
- High-pass filter ()

Process:

- Neighbourhood Scrutiny: The filter examines each pixel in relation to its surrounding pixels.
- Difference Emphasis: It highlights areas where the intensity values of a pixel differ significantly from its neighbours.
- Sharpness Boost: This emphasis on differences creates a sharpening effect, enhancing edges and fine details.



Original Image (Fig.1)

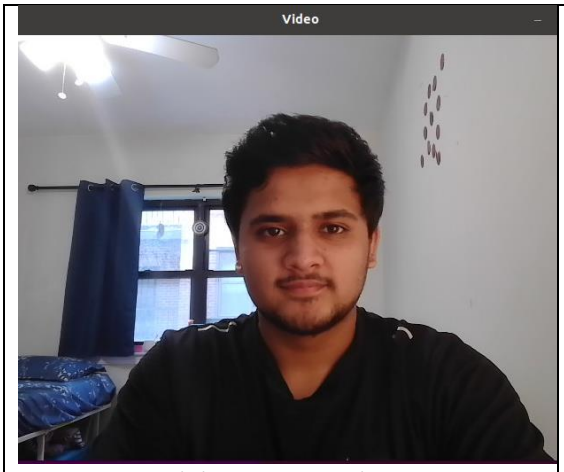


High-pass filter (Fig.2)

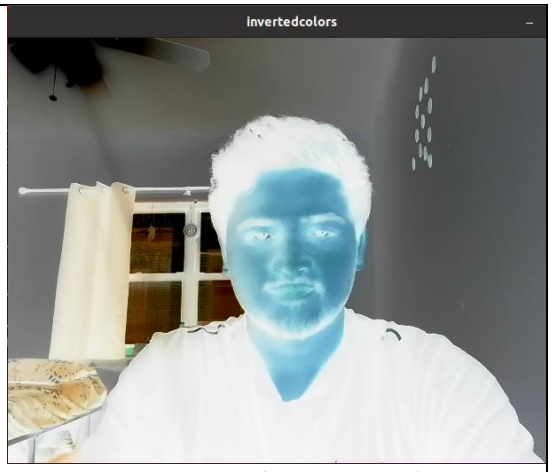
- Color Inversion

Process:

- Pixel by Pixel: The algorithm iterates through each pixel in the image.
- Channel Flip: Within each pixel, the red, green, and blue (RGB) values are individually reversed. Black becomes white, red becomes cyan, and so on.
- New Image: This flipped color data is then used to create a new image.



Original Image (Fig.1)



Inverted (Negative effect) (Fig.2)

Learning Outcomes:

Through the Video-special effects project, we gained a comprehensive understanding of the C/C++ OpenCV package and its applications in image processing. The project guided us from basic image reading and display functionalities to advanced tasks such as live video streaming, face detection, and the implementation of various filters and effects (i.e sepia tone filter, canny edge, vignetting effect, color isolation) to name a few. The hands-on experience enabled us to manipulate and enhance visual elements creatively, reinforcing our knowledge of computer vision concepts.

Acknowledgement:

We would like to express our gratitude to Medium.com, ChatGPt, and StackOverflow for providing valuable insights into filters and calculations, which served as a helpful reference throughout the Video-special effects project. The OpenCV tutorials laid a strong foundation for our work, while various YouTube videos offered practical insights that significantly contributed to the successful completion of the project. I am truly thankful for these resources, as they not only facilitated our learning but also made the implementation process much smoother.