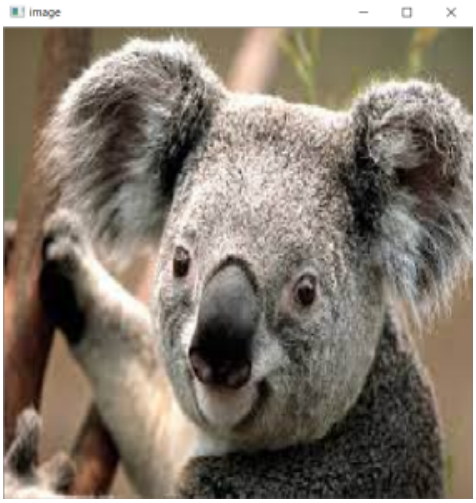
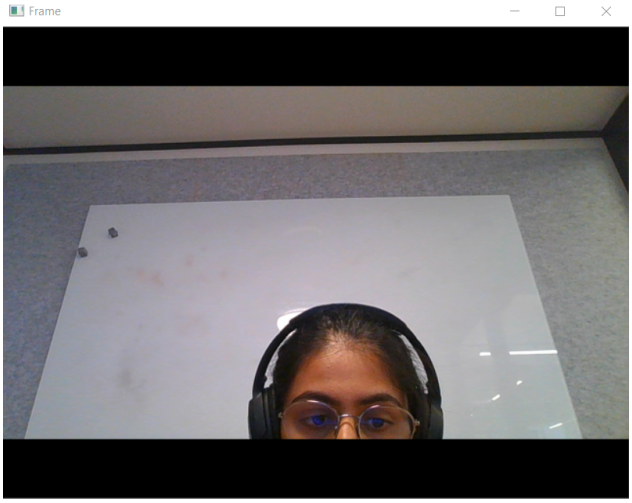
Project 1: Real-time filtering

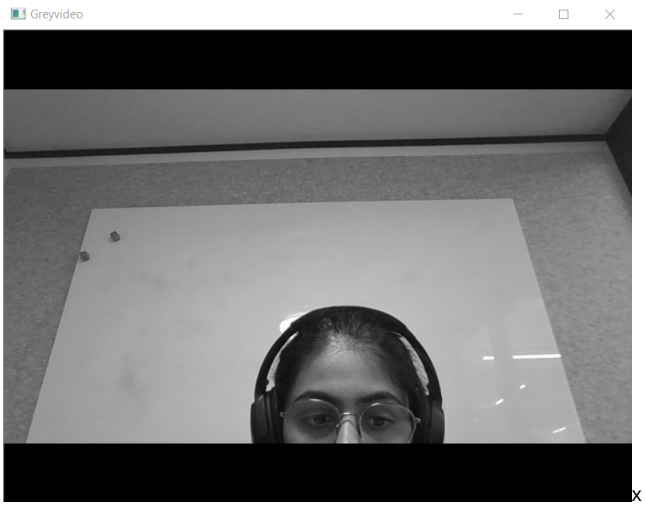
1. In ImgDisplay.cpp, it reads an image file and displays it in a window. It enters in the loop and checks for the keypress event, if the user presses 'q', the program quits.



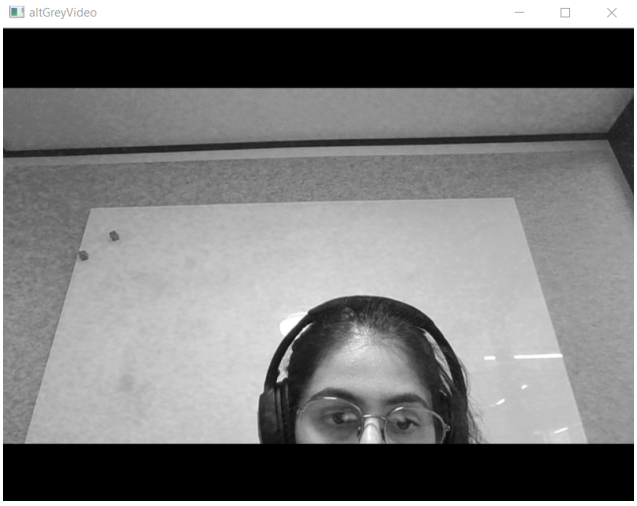
2, 3, 4. In vidDisplay.cpp, its main function opens video channel, creates window then goes into loop and check keypress event. if keypress event gets 's', it will save image, 'g' for greyscale, 'h' for alternate greyscale and finally if keypress event get 'q', program quit.

The output images are as follows:



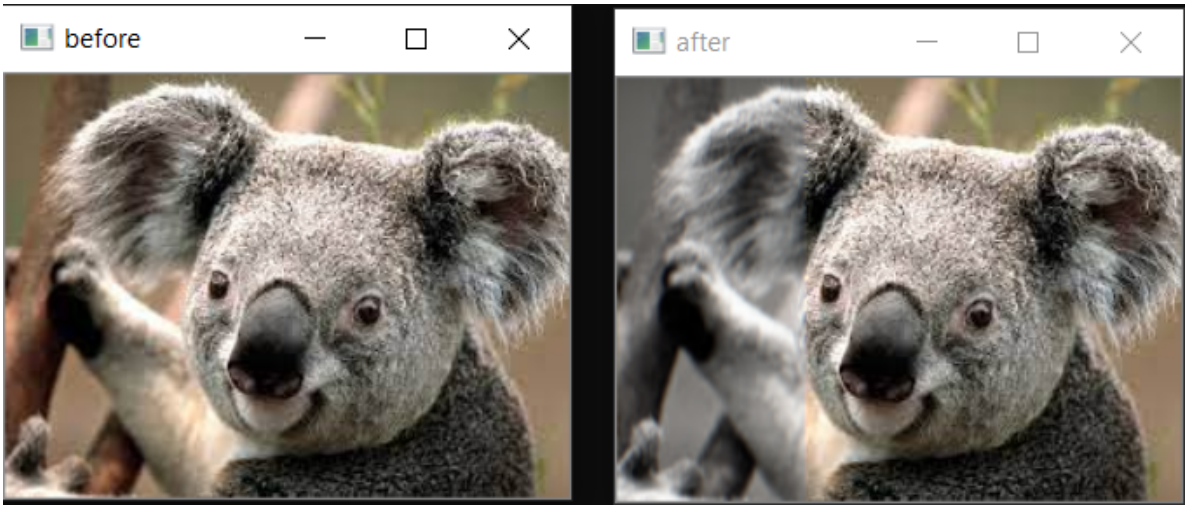


x

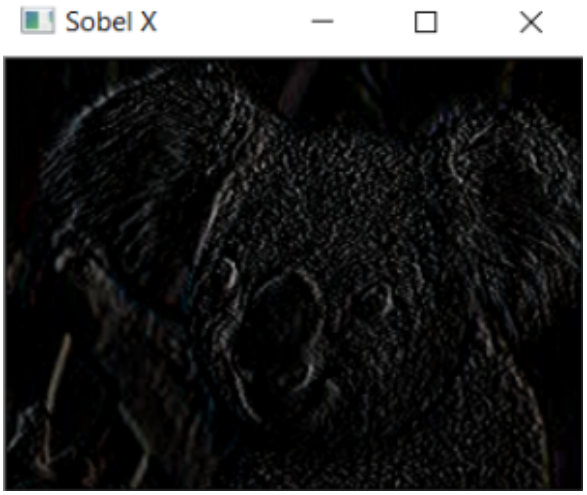


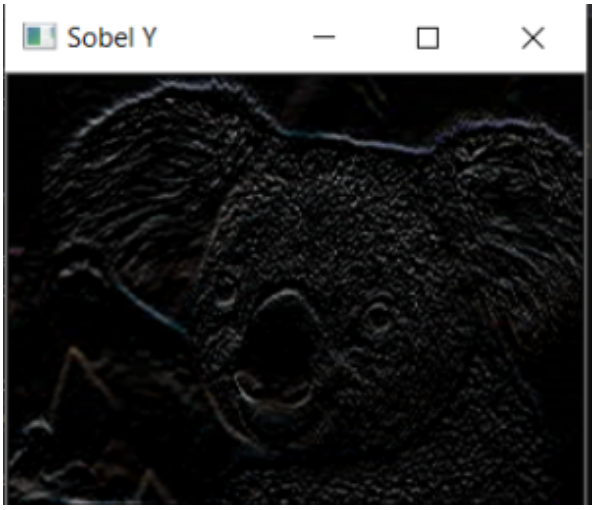
4. Alternate Greyscale video is computed by converting the colorspace of BGR to HSV.

5. In filter.cpp, we have created a method 'blur5x5', which shows a 5x5 Gaussian filter as separable 1x5 filters ([1 2 4 2 1] vertical and horizontal). Below is the output image (before and after blurring the image in **color**).

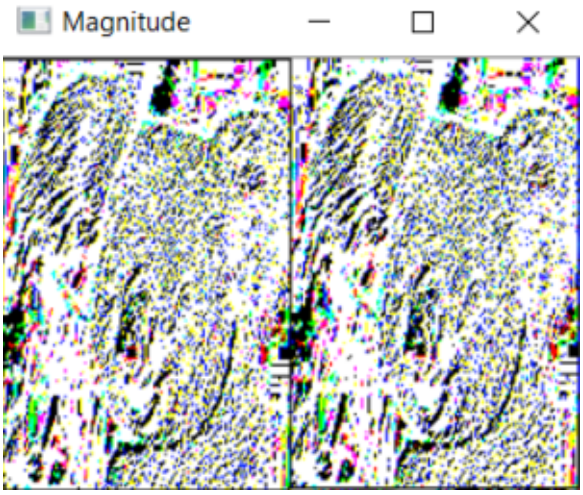


6.In filter.cpp, I have created a method 'sobel', which shows a 3x3 Gaussian filter as separable with x and y parameter vertical and horizontal.

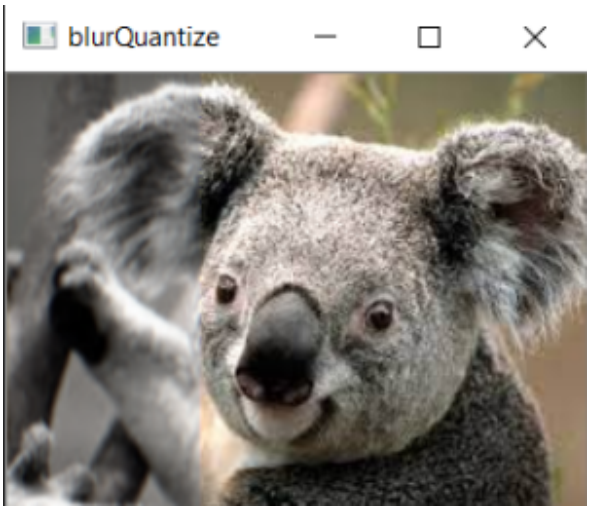




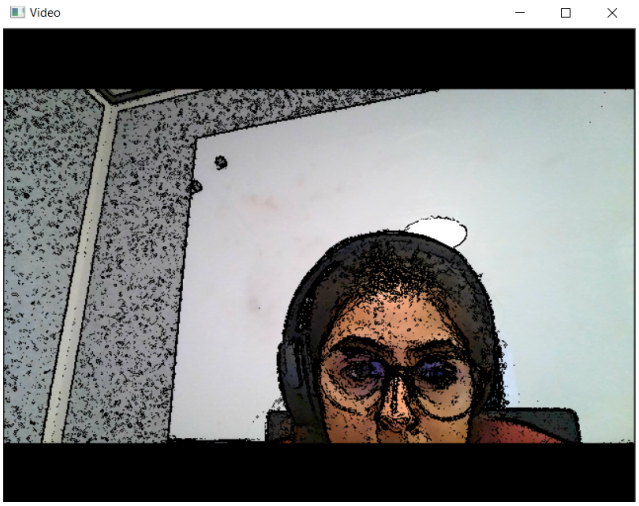
7. I implemented a function that generates a gradient magnitude image using Euclidean distance for magnitude: I = sqrt(sx*sx + sy*sy).



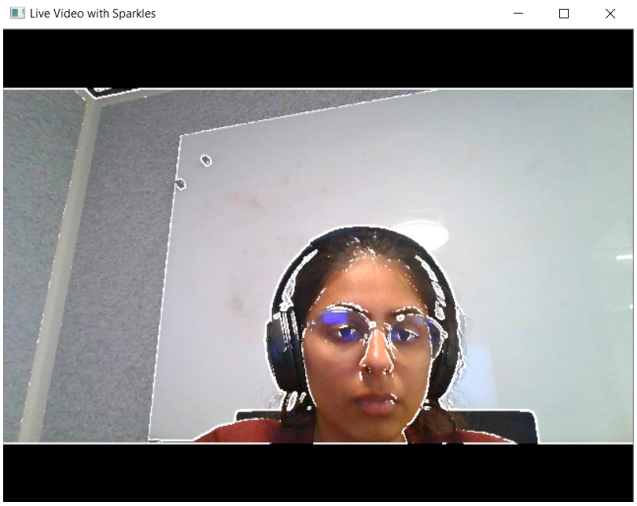
8. In filter.cpp, we create a function that takes in a color image, blurs the image, and then quantizes the image into a fixed number of levels as specified by a parameter, where the parameter levels is set to 15.



9. In filter.cpp, I created a function that cartoonizes a color image. it first calculates the gradient magnitude. Then blur and quantize the image. and display like cartoon.



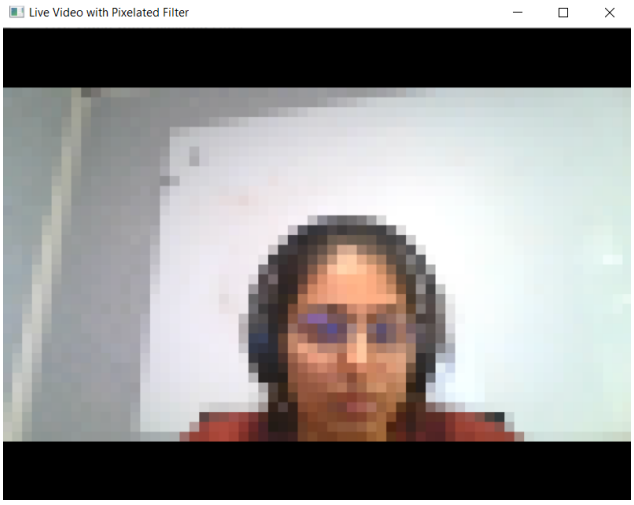
10. In filter.cpp, I created a function addSparkles that implements a video stream and adds sparkles into image (on the outline only) on key press 's'.



```
void addSparkles(const cv::Mat& src, cv::Mat& dst, int threshold)
{
    cv::Mat gray, sobelX, sobelY, mag;

    cv::cvtColor(src, gray, cv::COLOR_BGR2GRAY);
    cv::Sobel(gray, sobelX, CV_16S, 1, 0);
    cv::Sobel(gray, sobelY, CV_16S, 0, 1);
    cv::convertScaleAbs(sobelX, sobelX);
    cv::convertScaleAbs(sobelY, sobelY);
    cv::addWeighted(sobelX, 0.5, sobelY, 0.5, 0, mag);

    dst = src.clone();

    for (int row = 0; row < src.rows; row++)
    {
        for (int col = 0; col < src.cols; col++)
        {
            uchar magVal = mag.at<uchar>(row, col);
            cv::Vec3b& dstVal = dst.at<cv::Vec3b>(row, col);

            if (magVal > threshold)
            {
                dstVal = cv::Vec3b(255, 255, 255);
            }
        }
    }
}
```

11. Extension: Pixelated filter (on key press 'p' it outputs a live video with a pixelated filter)

```
void pixelateFilter(const cv::Mat& src, cv::Mat& dst, int blockSize)
{
    dst = src.clone();
    for (int row = 0; row < src.rows; row += blockSize)
    {
        for (int col = 0; col < src.cols; col += blockSize)
        {
            int blockRows = std::min(blockSize, src.rows - row);
            int blockCols = std::min(blockSize, src.cols - col);
            cv::Vec3d sum(0, 0, 0);
            int numPixels = 0;
            for (int r = 0; r < blockRows; ++r)
            {
                for (int c = 0; c < blockCols; ++c)
                {
                    sum += src.at<cv::Vec3b>(row + r, col + c);
                    numPixels++;
                }
            }
            cv::Vec3b averageColor = sum / numPixels;
            for (int r = 0; r < blockRows; ++r)
            {
                for (int c = 0; c < blockCols; ++c)
                {
                    dst.at<cv::Vec3b>(row + r, col + c) = averageColor;
                }
            }
        }
    }
}
```



Reflection

I learned quite a bit during my first openCV project. By working on this project, I learned how to Read and display images and videos using OpenCV and how to Implement custom filters and effects, such as blurring, edge detection with Sobel operators, and the pixelated effect. I have gained valuable hands-on experience in image processing and computer vision, and have improved your programming skills in C++ and OpenCV. These skills can be further expanded by exploring more advanced techniques, applying them to different projects, or optimizing my existing implementations for better performance.