CS5330_Project1

A short description of the overall project in your own words. (200 words or less)

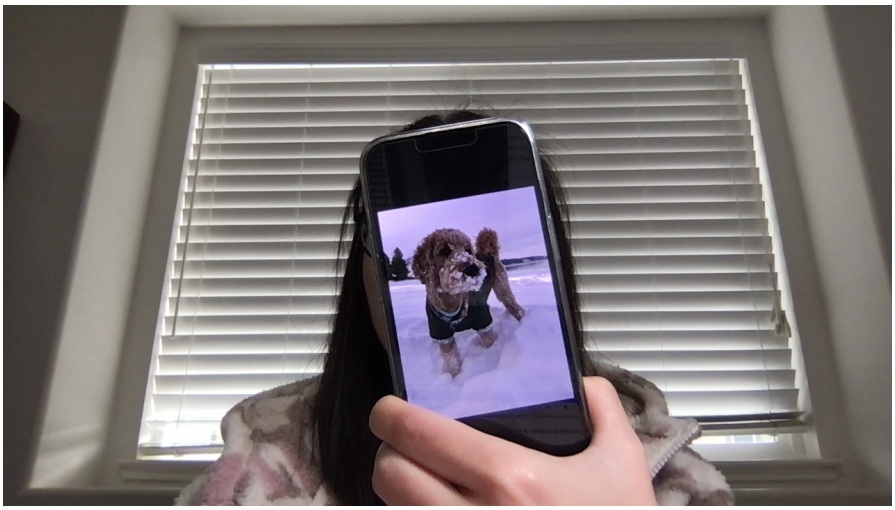
This project aims to help us get familiar with C++ and OpenCV, with a focus on image/video read, display, manipulate, and write.

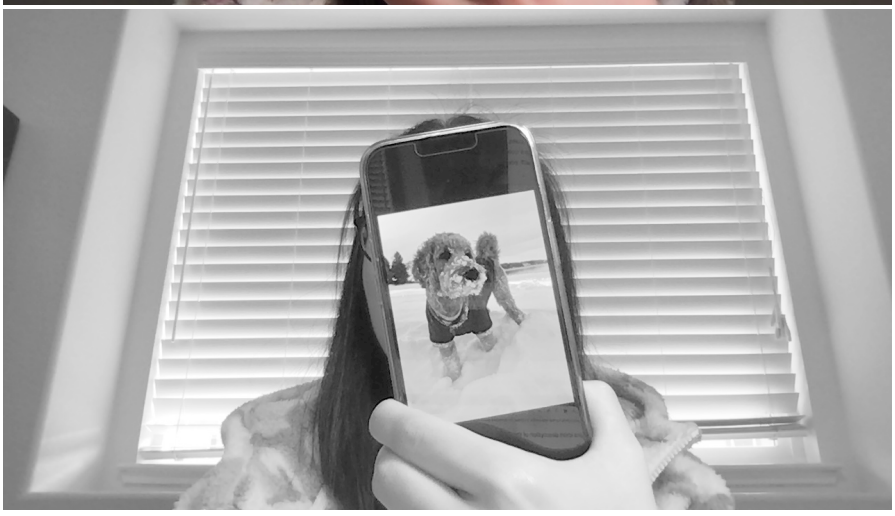
It requires us to implement functions to read images from a local file or read video streams from a live camera, display the live video, apply a variety of filters and effects to the live video, and save the image or video. It also requires us to get familiar with OpenCV built-in functions while understanding how to access and modify an image by accessing pixels directly.

Any required images along with a short description of the meaning of the image.

Required Image 1 - cvtColor grayscale

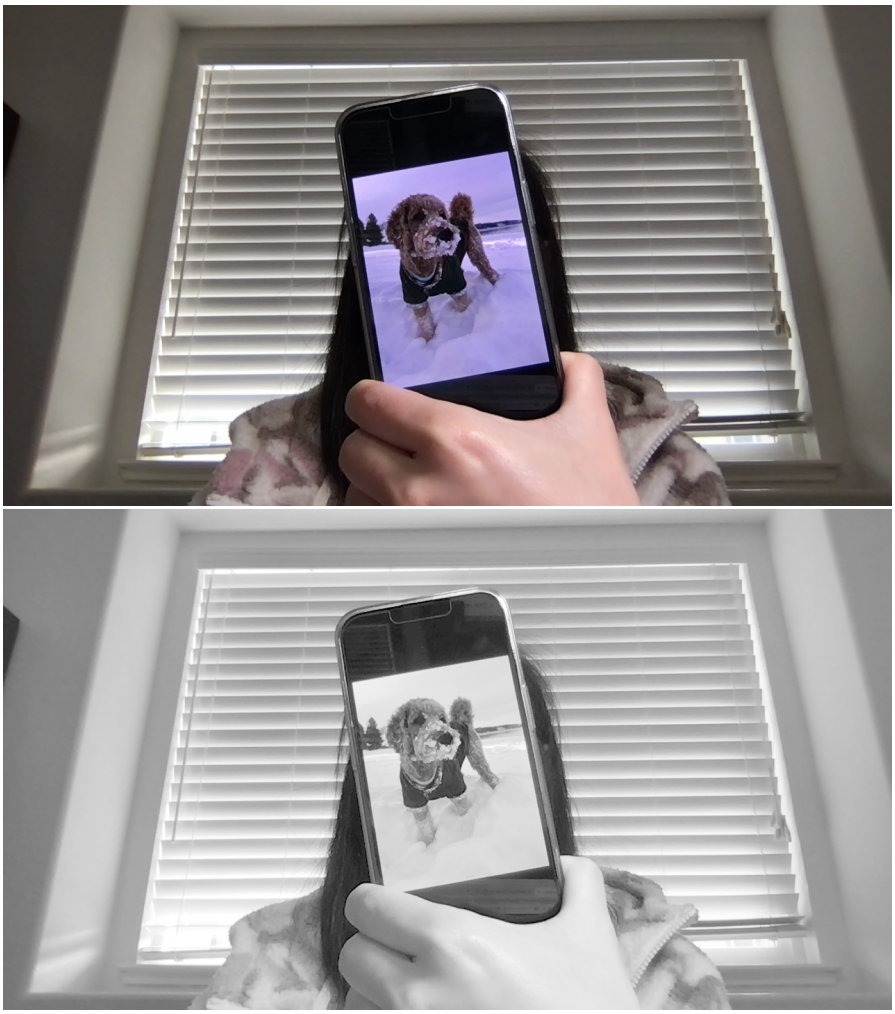
The weight of each color is 0.299 R + 0.587 G + 0.114 B.





Required Image 2 - customized grayscale

Compute the average value of the BGR channels and apply the result to the dst. (B + G + R) / 3



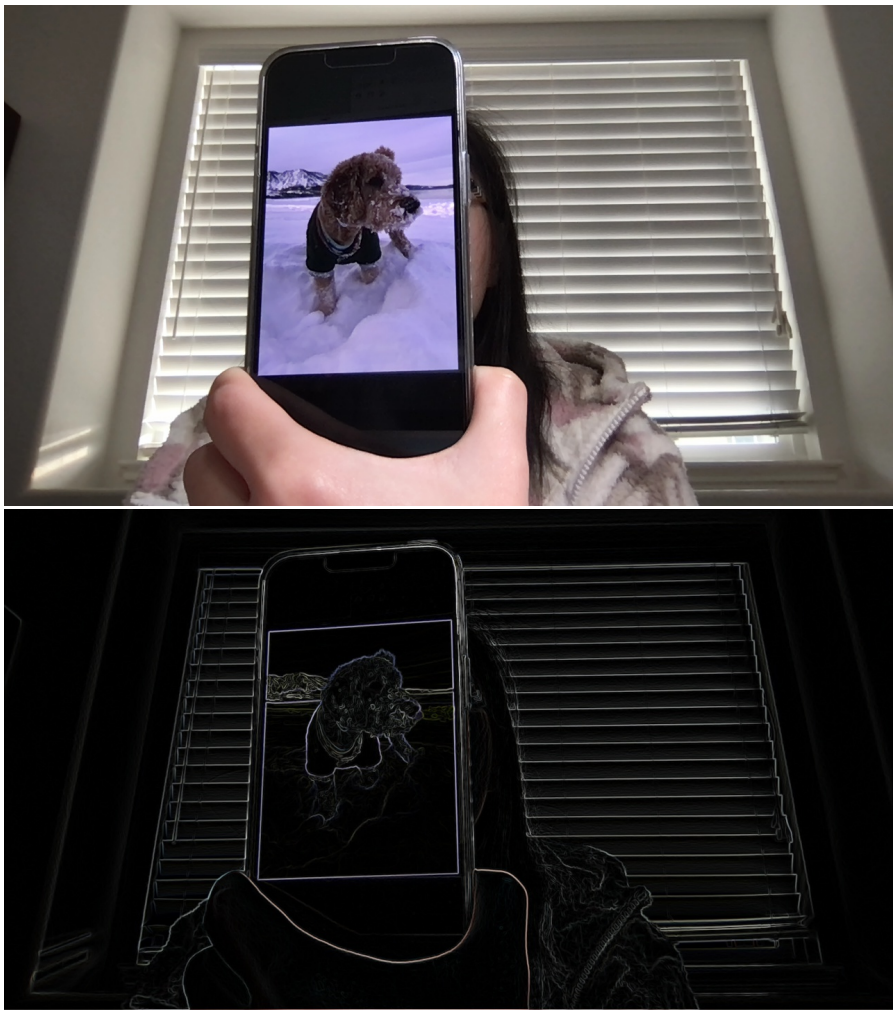
Required Image 3 - Gaussian blur

Apply Gaussian filter as separable 1 x 5 filters by accessing and modifying each pixel directly.

This and all the other self-defined functions treat pixels off the edge as having asymmetric reflection over that edge.

Required Image 4 - gradient magnitude

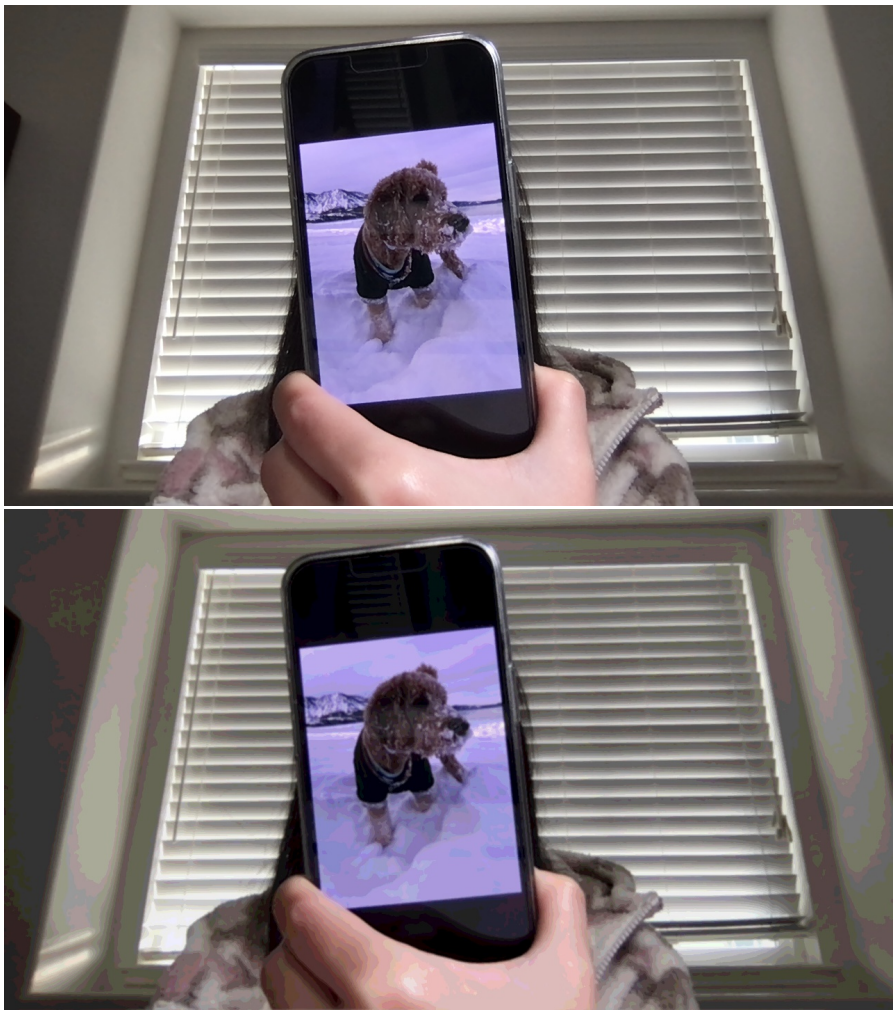
Apply SobelX and SobelY filters to the original image. Then calculate the gradient magnitude. This will display the edges in the image.



Required Image 5 - blurred and quantized

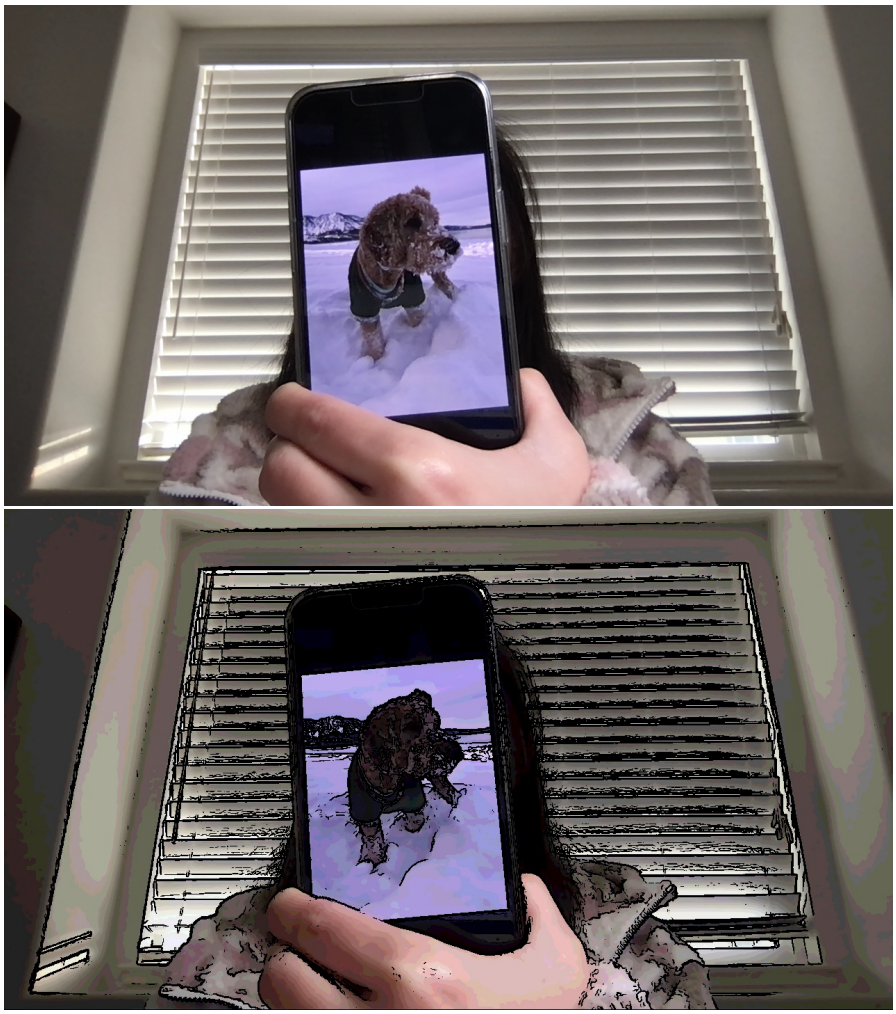
First, blur the image, then quantize it with a level of 15.

This will compress a range of similar values into a single quantum value. Therefore, in the image, it displays some color blocks.



Required Image 6 - cartoonized

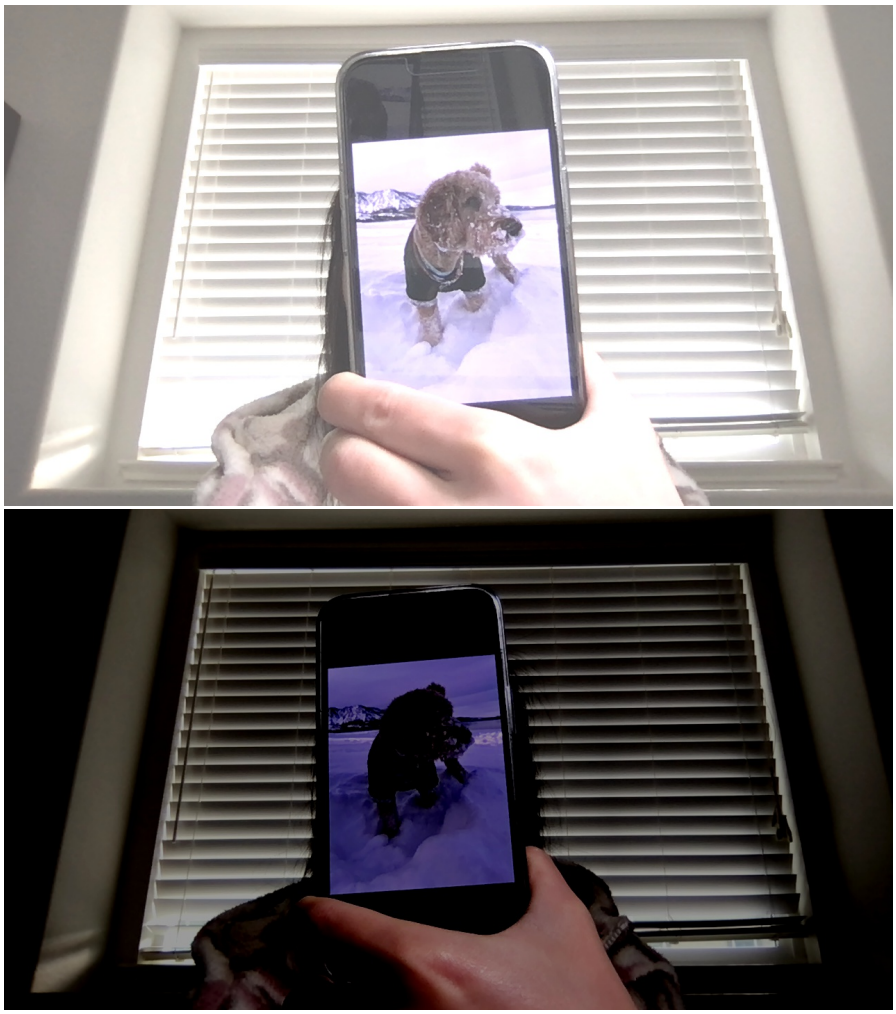
First, calculate the gradient magnitude of the original image, and save the data in a separate Mat. Then blur and quantize the original image(with a level of 15). Finally, set to black any pixels with a gradient magnitude larger than a threshold of 15.



Required Image 7a - adjust brightness

To increase brightness, we add some positive constant value to each and every pixel in the image. To decrease the brightness, we subtract some positive constant value from each and every pixel.

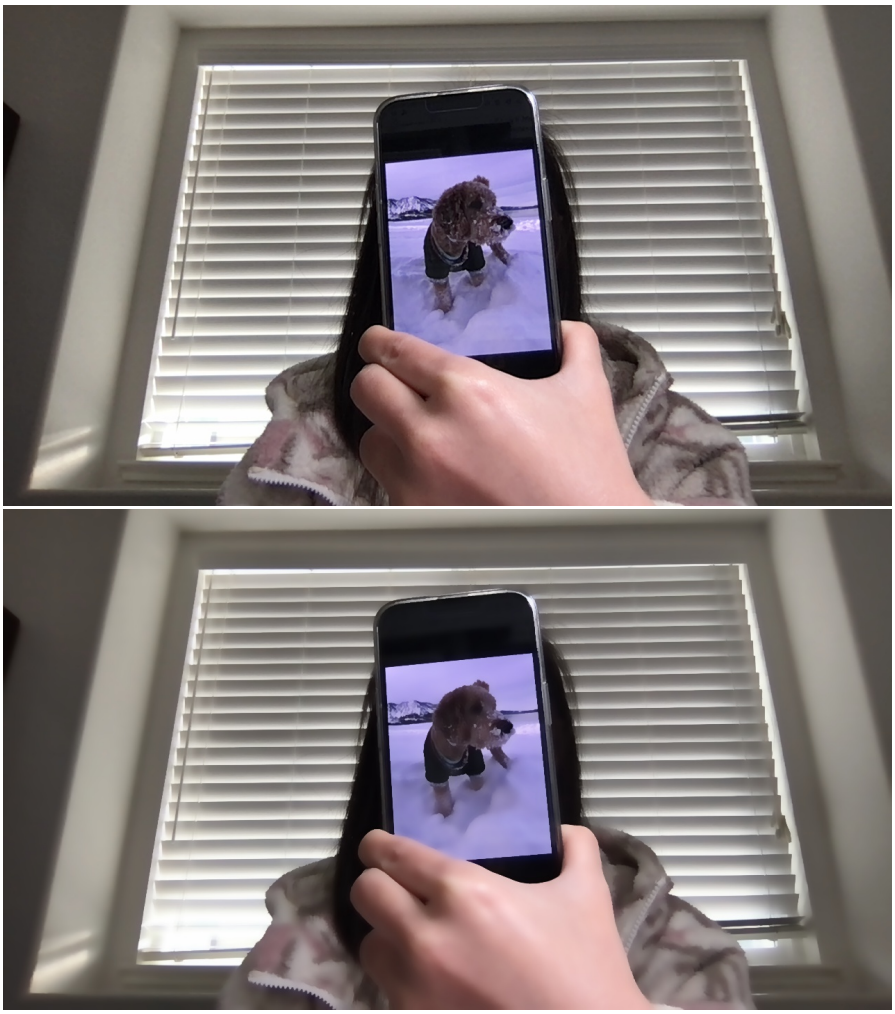
If the pixel value exceeds the min or max allowable limit after adding or subtracting the value, the min or max value should be used instead of the calculated value.



Required Image 7b - bilateral blur

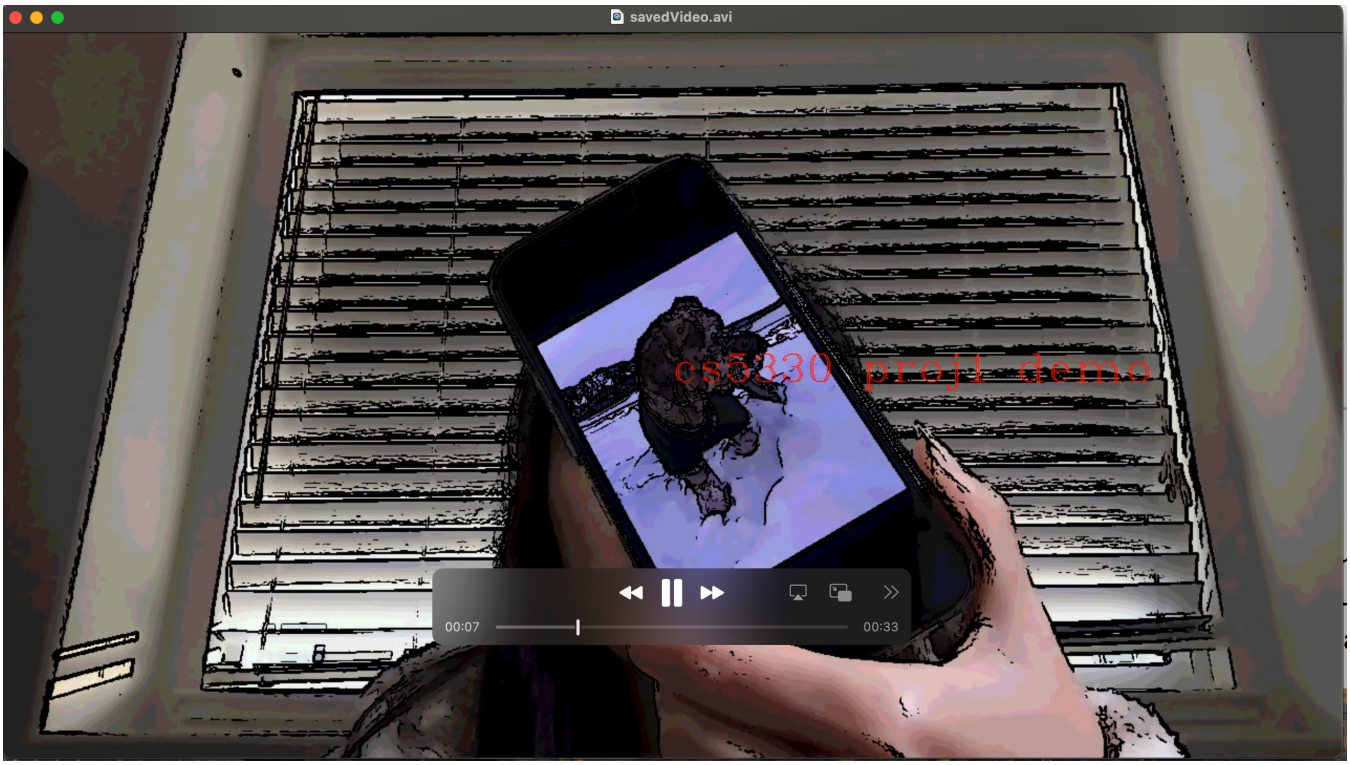
This is achieved by using the bilateralFilter() function from OpenCV.

The bilateral filter replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. It can reduce unwanted noise very well while keeping edges fairly sharp.



A description and example images of any extensions.

Link to an example of the saved video(with meme and different effects): https://drive.google.com/file/d/1SBnRlHvLVfztqg9cmsJL5NudX_1_epCI/view?usp=sharing



Extension 1 - add meme to saved image or video

When users want to save an image or a video sequence, the program will ask them to input a meme. The meme will be added to the image/video through putText() method from OpenCV.

Extension 2 - save video sequence

Users will be able to save a video sequence with different effects. I used the VideoWriter object from OpenCV. There are two major difficulties.

First, since the VideoWriter object requires 3-channel images, we need to convert any 1-channel(greyscale) images to 3-channel before writing them to the video stream. This is achieved by cvtColor(src, dst, COLOR_GRAY2BGR).

Second, some effects(e.g. cartoon, magnitude) are more time-consuming than others, therefore, there will be a very fast playback for the video clip with those effects. I believe this is because the processing time for those effects can not meet the FPS requirements of the VideoWriter object. I tried to lower the FPS in the VideoWriter object, however, with a lower FPS the original video or video with faster effects will be played very slow. Then after some research, I believe it is a good practice to replicate the frames of the slower effects, so more frames will be added and there won't be any fast playback.

To calculate a proper number of replicated frames needed, I measure the elapsed time of capturing a frame(without applying any effects) and set it as the base time. I also measure the elapsed time of applying each filter/effect to the frame. The number of replicated frames N = (base time + effect time) / base time. Then every time a frame is written to the video, the program will call VideoWriter.write(frame) N times and N replicated frames will be added to the video.

A short reflection of what you learned.

I learned a lot from this project. On a high level, computer vision is a completely new field to me, this first project gives me a better insight into it.

To be more detailed, this is my first time writing c++ and this project familiarized me with the basic usage of c++ and cmake. I also learned how to use OpenCV to work on images. Also, it was fun to learn about how different filters and effects actually work. One big challenge is to understand the types of different image frames. I spent quite a lot of time trying to figure out why my filter1xN() and filterNx1() didn't work as expected before I realized it was related to the type of the src and dst.

Acknowledgment of any materials or people you consulted for the assignment.

- OpenCV Tutorials https://docs.opencv.org/4.5.1/index.html
- Learn OpenCV https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
- How to use Chrono library https://www.techiedelight.com/measure-elapsed-time-program-chrono-library/

- Change brightness using openCV https://www.opencv-srf.com/2018/02/change-brightness-of-images-and-videos.html