**A Short Report on the Project**

First of all, we imported a library called OpenCV into our program and assigned it the alias cv. We chose this library because all the tasks given in the assignment can be accomplished easily with it.

After that, we used the imread function of OpenCV to read an already stored image and passed the path of that image as a parameter, then stored that image in a variable named frame.

To ensure that the image was read correctly, we wrote an if statement that generates an error message if the image is not found in the frame variable. This prevents errors later in the program.

Next, we resized the image using the resize function of OpenCV to standardize the input size. The resized image is stored in a variable named frame\_resized. The target resolution is 640x480 pixels, as required in the assignment.

We then converted the image to grayscale using the cvtColor function. The function takes two parameters: the variable holding the image we want to change and the COLOR\_BGR2GRAY conversion code, which converts the color image to grayscale. The grayscale image is stored in a variable named frame\_bw. Converting to grayscale helps in reducing computational complexity and focusing on structural features.

After converting to grayscale, we applied a noise reduction technique using the medianBlur function. Instead of applying noise reduction to the colored image, we applied it to the grayscale image to enhance edge detection results. The median blur filter helps in reducing noise, particularly salt-and-pepper noise, while preserving edges. The processed image is stored in a variable named frame\_denoise.

For edge detection, we used the Canny edge detection algorithm. We chose Canny over Sobel because Canny is more efficient, provides cleaner edges, and reduces noise effects. Unlike the initial implementation where edge detection was applied to the resized image, we applied it to the noise-reduced grayscale image to improve accuracy. The detected edges are stored in a variable named frame\_edges.

Finally, we used the imwrite function to save all processed images, including the resized, grayscale, noise-reduced, and edge-detected images, to disk. We also displayed them using the imshow function, which shows each processed image in a separate window. The program waits for the user to press the 'd' key before closing the displayed images using cv.waitKey(20).