SOBEL EDGE DETECTION (PHASE 1)

Alexander Robles, Khadichabonu Valieva

CSC 410

University of Southern Mississippi

School of Computing Sciences and Computer Engineering

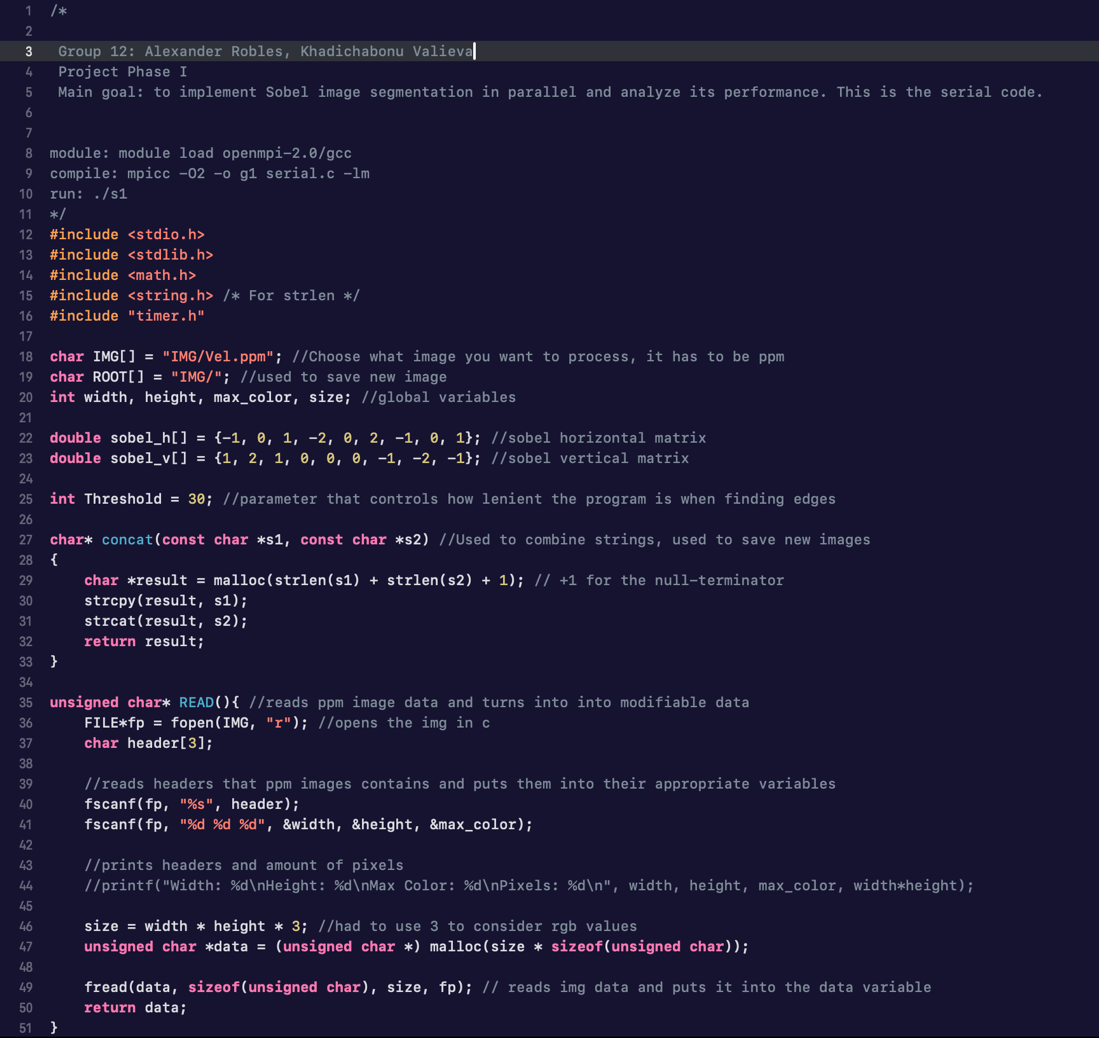
SOBEL EDGE DETECTION (PHASE 1)

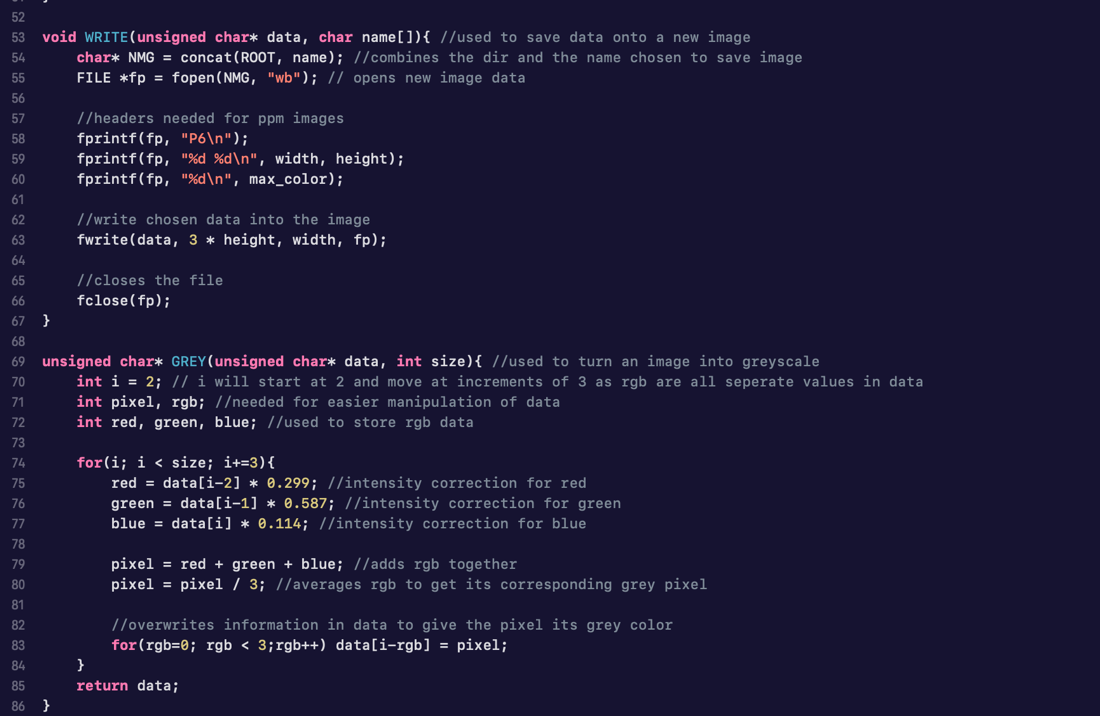
**Problem Description, Methods, and Techniques**

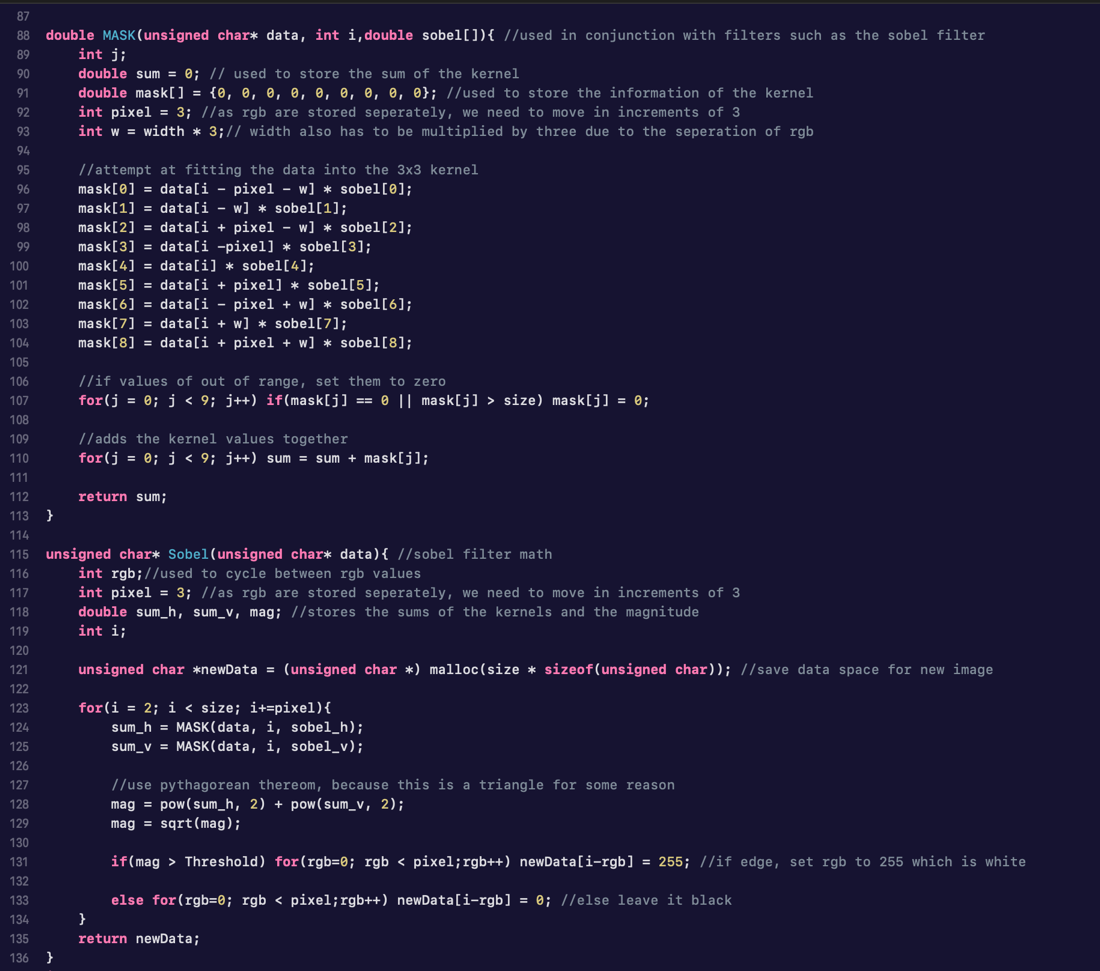
“The project aims to implement Sobel image segmentation in parallel and analyze its performance. The Sobel operator will be employed to identify edges within the image, which will serve as the basis for segmentation.”

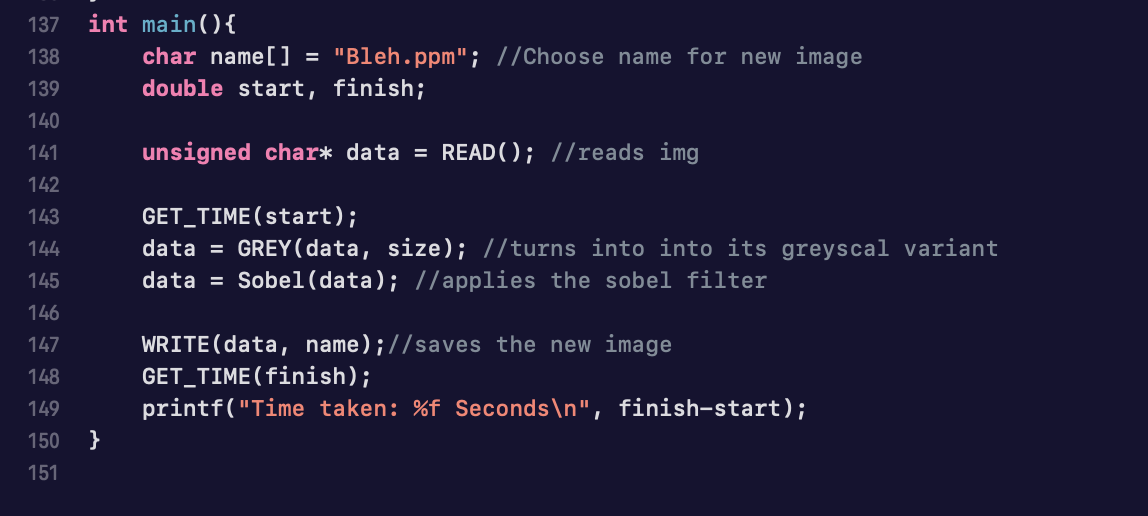
For this project we used convolution filters, which is a 3x3 matrix repeated over every pixel in the image using the Sobel matrixes used to compare surrounding pixel RGB values to tell if the current pixel is an edge or not. We also have included the timer.

**Serial implementation using C and its results**

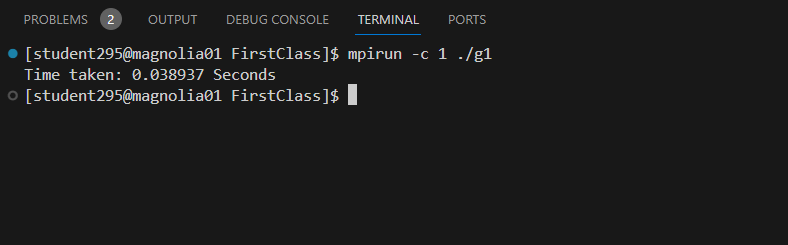






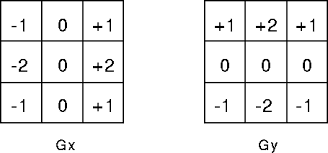


Result:



**Design of parallel algorithm**

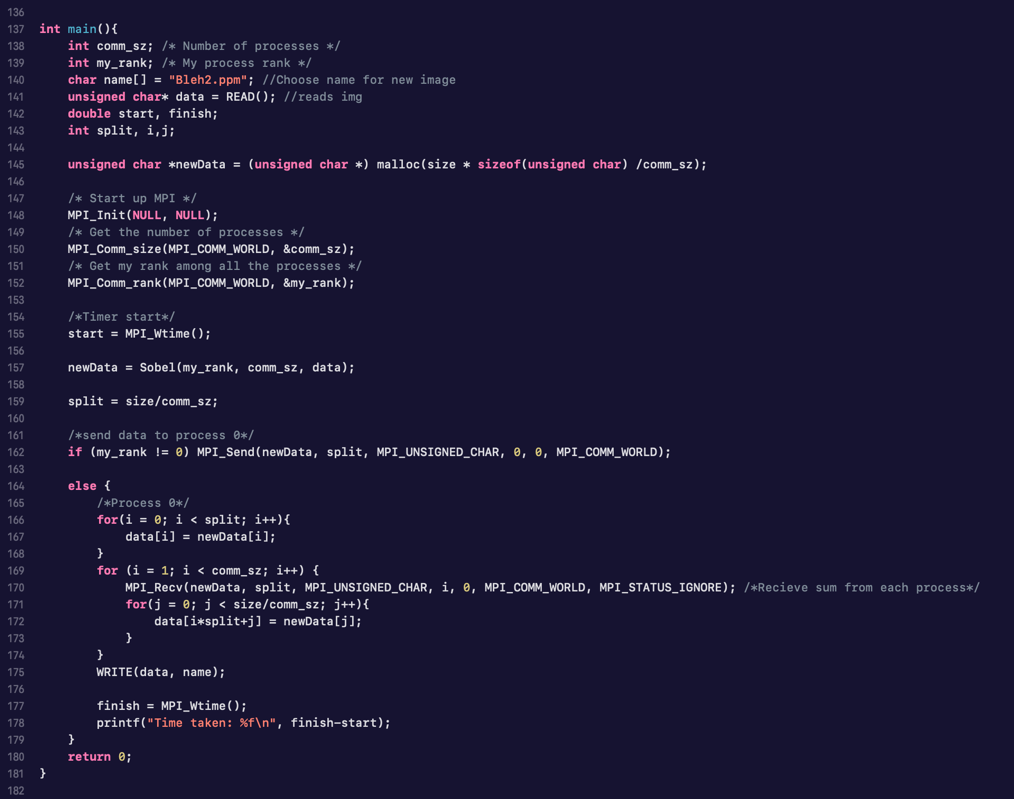
Sobel operator is a discrete differential operator, it computes the gradient approximation of image intensity function for image edge detection. We used 3x3 kernels/masks to calculate the vertical and horizontal Sobel matrices:

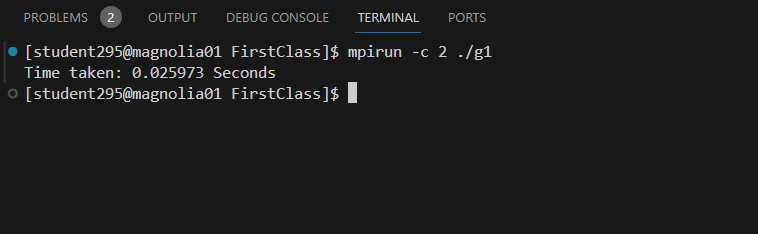


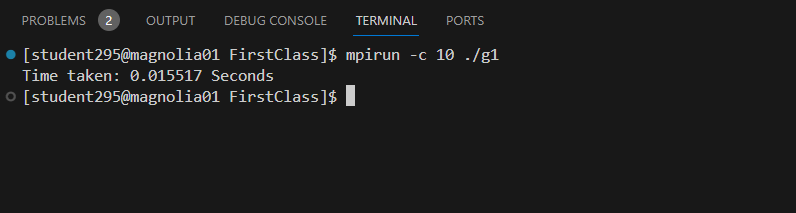
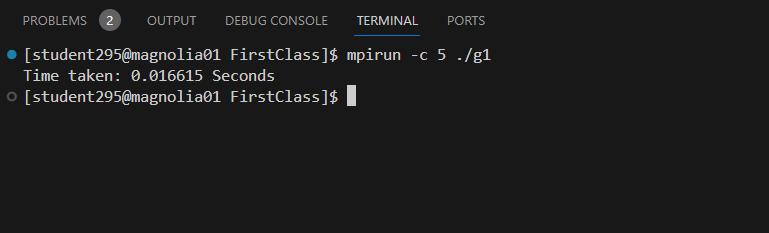
The process included reading an image, converting the actual RGB image to the grayscale image, then we use MASK function that applies a Sobel filter to a pixel of the image, as well as Sobel function that calculates the Sobel edge detection for a portion of the image based on the rank of the process and comm\_sz. Finally, in the main function, we initialize MPI, use basic MPI commands, read the image, execute Sobel function in parallel based on the rank and size, and finally, splits the image among processes and sends it to process 0 where it gathers all data and rebuilds the final image.

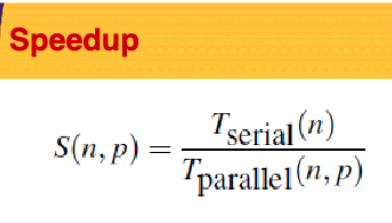
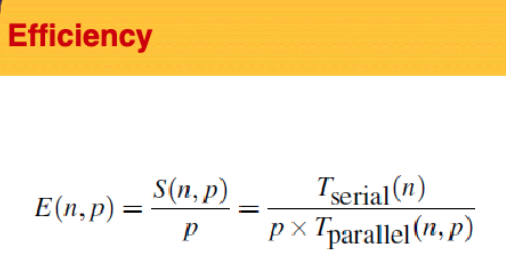
**Implementation of the parallel algorithm using MPI**

**[-almost all code lines before, so we are including screenshot of only the main function-]**

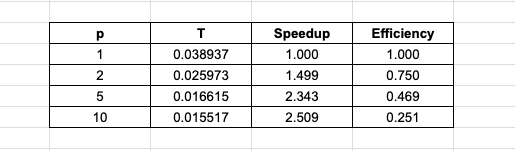


**Compilation and execution of your program on the Magnolia system. Include commands, screenshots showing your account, compilation, execution, and performance metrics (runtime and speedup)**



For evaluation, we calculated the speedup and efficiency according to the following formulas:  
 

And got the following results:



**Comparison of output images obtained from the serial and parallel programs.**

**Performance analysis and discussion of the parallel implementation, including tables and figures.**

As the table shows, with the increase of number of processes, the speedup also increases, which justifies the purpose of using parallel programming. However, as we see the results for efficiency, the percentage decreases with the increase of processes. It suggests that the communication between processes also increases and becomes significant to the computation time, which is why the parallel program may not be scaling well.

**Conclusion**

In this project, we were able to learn using Soble image segmentation in parallel and implement it in practice. We were able to successfully get the image and compare it with the original RGB version from serial and parallel programs. Finally, we showed the performance analysis in the form of the table that includes time it took for different number of processes to run with parallel, their speedup as well as efficiency compared to the serial code.