Parallelizing Sobel Edge Detection and Gaussian Blur Algorithms

Sumanth N Hegde
Mechanical Engineering
National Institute of Technology
Karnataka, Surathkal
sumanthnhegde.191me284@nitk.edu.in

Pratham N
Metallurgical and Materials Engineering
National Institute of Technology
Karnataka, Surathkal
pratham.191mt034@nitk.edu.in

Saransh Bhaduka
Mechanical Engineering
National Institute of Technology
Karnataka, Surathkal
saranshbhaduka.191me175@nitk.edu.in

Abstract—**This work presents the parallelization of commonly used image processing algorithms like Gaussian blur and Sobel edge detection using OpenMP and MPI. Gauss blurring is a well-known image processing technique that reduces image noise and detail. In contrast, edge detection is one of the fundamental tasks in image processing used for object identification, segmentation, and robot vision. Dealing with images, the algorithm inherits repetitive instructions that depend on the image size, thus may slow down the processing speed. Single-threaded applications cannot catch up with parallel systems when it comes to scalability mainly because of the clock frequency limitation of modern CPUs and economic reasons. Using shared memory or distributed memory architectures, parallel systems can provide tremendous speed up compared to single-threaded systems. The test is being done on images of varying sizes of 300x300, 1000x800, and 1920x1080 pixels.**

Keywords—Parallel programming, Sobel edge algorithm, Gaussian Blur, OpenMP, Message Passing Interface

I. Introduction

An image can be defined as a two-dimensional function, f(x, y), where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or grey level of the image at that point. Image Processing is a method to convert an image into digital form and perform some operation to get an enhanced image or extract some useful information from it. It is a type of signal dispensation in which input image, like a photograph or video frame, and output may be image or characteristics associated with that image. Usually, an image processing system includes treating images as two-dimensional signals while applying already set signal processing methods to them. Image processing has been developed for two purposes. The first is for better interpretation of images by humans, and the second is for analysis, data storage, and transfer. Significant improvements have been made to image processing as it is widely being used in many areas such as security, medical imaging, surveillance, retrieval, etc.

With the advancement in image capturing machines and other technology, image size has increased, and hence, necessary techniques have to be adopted to reduce the processing time and one such method to accomplish this is by using parallelization. In general, parallelism refers to simultaneously performing the computation by two or more computing units. Parallel computing can be done using parallel processing equipment or with the help of distributed computing system. The first approach can be made in a computer system having two or more processors, and in the other approach, many interconnected computers are used to accomplish the computation task.

The selection of the architecture is based on the behaviour of the application and the involved budget. For decreasing the time consumption, parallel paradigms such as CUDA, MPI, and OpenMP are used. OpenMP is a library used for shared memory architecture, whereas MPI or the message passing interface is used in distributed computing. CUDA, which stands for Compute Unified Device Architecture, is a library provided by NVIDIA that enables developers to speed up compute-intensive applications by harnessing the power of GPUs for the parallelizable part of the computation. The paper is organized as follows. Image processing algorithms like Gaussian blur and Sobel edge detection have been presented in sections 2 and 3, respectively. Finally, the computation time for using different parallelization techniques is compared using several graphs and discussed in the results section.

II. Sobel Edge Detection

A. *What is Edge Detection and why is it important?*

Edge detection is a widely used image segmentation technique that identifies edges in an image. The edges are also represented as a collection of interconnected points located at the border between the two areas. The edges illustrate the borders of the object, and hence this feature can be used to segment an image toward its primary areas or objects. In general, edge detection aims to significantly reduce the amount of data in an image while preserving the structural properties for further image processing. Fingerprint identification, finding pathological objects in medical images such as tumors, optical character recognition, and face detection are some fields in which edge detection is used. There are several edge detection algorithms in image processing, such as Canny, Sobel, Prewitt, Robert, and in this paper, we have discussed the implementation of Sobel edge detection.

B. *Working of Sobel Edge Detection*

Presented as a discrete differential operator technique for gradient approximation computation of the image intensity function, Sobel edge detection enables low-level feature extraction and dimensionality reduction, essentially reducing noise in the image. It has been handy in facial recognition applications.

Generating the lower dimension output requires us to take the derivative of the image. First, we calculate the derivate in both the x and y directions. Then we create two 3x3 kernels, with zeros along the center of the corresponding axis, twos in the center squares perpendicular to the central zero, and ones in each corner. These kernels will then convolve over the image, placing the central pixel of each kernel over each pixel in the image. Matrix multiplication is used to calculate the intensity value of a new corresponding pixel in an output image for each kernel and, hence we end up with two output images. The aim is to find the change or difference between the pixel in the image and all pixels in the gradient matrix.

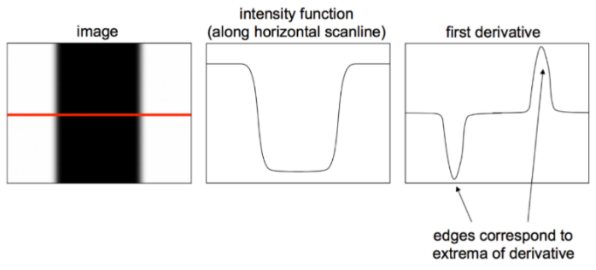

Figure 1: Working of Sobel edge detection

The calculated values for each matrix multiplication between the mask and 3x3 section of the input are summed to produce the final value for the pixel in the output image. This generates a new image that contains information about the presence of vertical and horizontal edges present in the original image. This is a geometric feature representation of the original image. Further, because this operator can produce the same output each time, this technique allows for stable edge detection for image segmentation tasks. The working of the sobel edge detection and the X, Y direction kernel has been shown in Figure 1 and 2 respectively.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 1: X, Y Direction Kernel

III. Gaussian Blur

In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). It is a widely used effect in graphics software, typically to reduce image noise and reduce detail..

Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales. A type of low-pass filter, Gaussian blur smoothens uneven pixel values in an image by cutting out the extreme outliers.

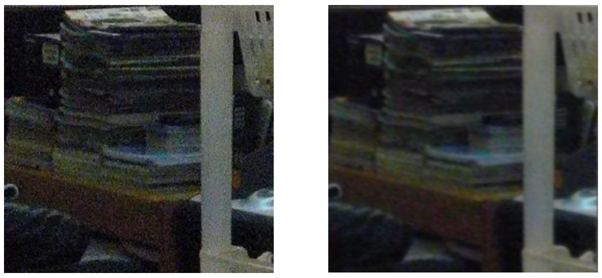

Figure 3: Night-sight noisy image rendered smooth by Gaussian Blurring

A. Uses of Gaussian Blurring

A halftone image can be rendered smooth and better looking using Gaussian Blur, as shown in the figure. If you take a smartphone photo in low light and the resulting image has a lot of noise, Gaussian blur can mute that noise as shown in Figure 3.

The formation of moiré pattern aliasing artifacts can be avoided as well during Gaussian Blurring. Even though the image is blurred, the aliasing is avoided. This is seen in the figures below.

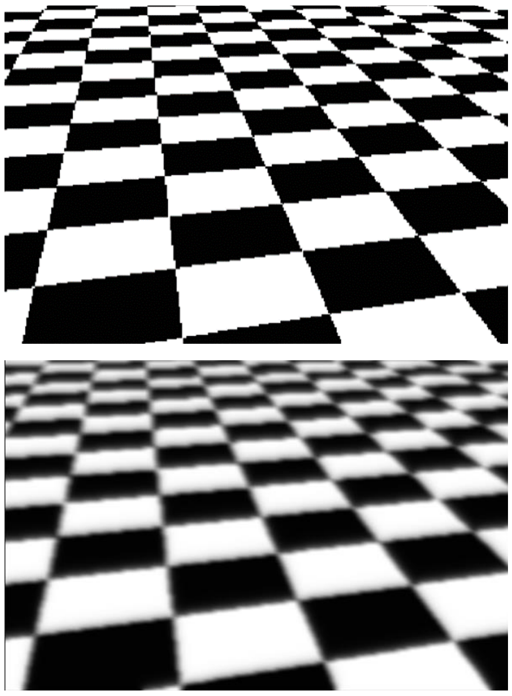

Figure 4: Moiré Pattern removed in the figure due to Gaussian Blurring

Gaussian smoothing is commonly used with edge detection. Most edge-detection algorithms are sensitive to noise. Using a Gaussian Blur filter before edge detection aims to reduce the level of noise in the image, which improves the result of the following edge-detection algorithm.

B. Convolution - In Simple Terms

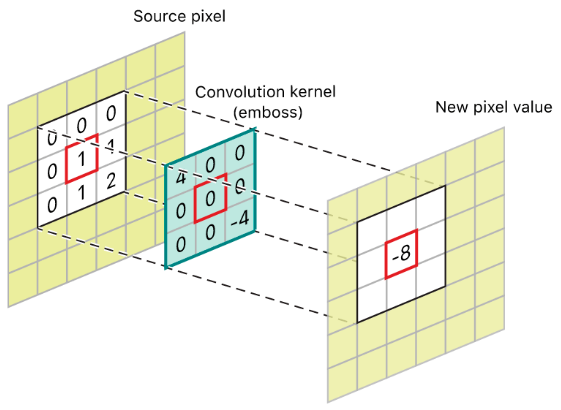

Figure 5: Convolution Kernel

In simple terms, convolution is simply the process of taking a small matrix called the kernel and running it over all the pixels in an image. At every pixel, we'll perform some math operation involving the values in the convolution matrix and the values of a pixel and its surroundings to determine the value for a pixel in the output image. By changing the values in the kernel, we can change the effect on the image — blurring, sharpening, edge detection, noise reduction, etc. This is seen in Figure 4.

C. *Gaussian Blur - the intricacies*

Two dimensional Gaussian functions are shown in Figure . Blur pixel has the peak value when it is compared with other neighbour weighted average pixels. The figure tells us that the maximum distance is five pixels for a picture and it is divided into equal parts. Thus, one can conclude that the window size is ten which is also called kernel size.

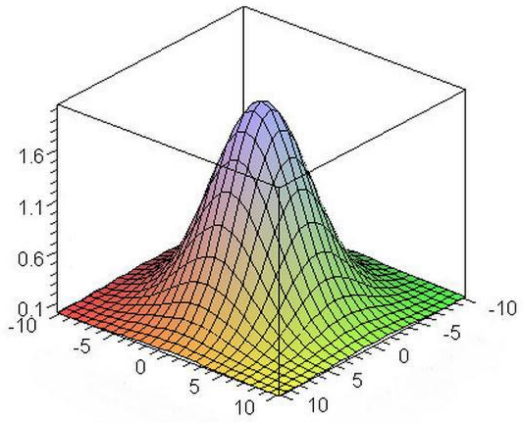

Figure 5: The graphical representation of the 2-dimensional Gaussian function

The weight of main pixel and the other weights of less weighted pixels can be calculated as in (1) Gaussian equation

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}} \qquad (1)$$

In this equation the parameters are explained as follows:

- σ : blur factor: If this value increase, image will blur.
- e : euler number
- x : Horizontal distance to centre pixel
- *y: Vertical* distance to centre pixel

According to this equation, x and y distances will be zero for centre pixel. When distance increases from centre pixel x^2+y^2 value will increase and weights will decrease. If this formula is applied to two dimensions, bell shape distribution appears from contour centre point and forms homocentric circle surfaces. These distribution values are used for making a convolution matrix. That matrix is applied to original image. Every new value of pixels can be computed by weighted averaging of itself and neighbour pixels. Centre pixel takes highest average weight. Nonetheless, the weight of the neighbour pixels is directly proportional to the distance to the centre. This process comes to a conclusion with a blur, conserves borders and edges.

IV. RESULTS AND CONCLUSIONS

The sequential Sobel edge detection algorithm and Gaussian Blurring was implemented in C++, with parallelization using OpenMP and MPI. OpenMP's multi-threading fork-join model was used to fork several slave threads and separate the task among them. The separated tasks are then performed simultaneously, with a run-time environment assigning threads to distinct tasks. The segment of code intended to run in parallel is stamped likewise with a pre-processor order that is used to join the outputs of the processes in order after they finish the execution of their corresponding task. In contrast, MPI allows the programmer to take advantage of the massive chain of processors to perform any general-purpose computation. Since both the algorithms can be broken down into many parallel executions, MPI proves to be the best solution. The algorithm was tested using images of various sizes, and a comparison of the execution times was made. The graphs representing the same have been shown in Figure 6. The sequential code and OpenMP parallelization were done using the following system specifications for Sobel Edge Detection:
Processor: Intel(R) Core (TM) i5-4300U CPU @ 1.90GHz
Memory: 8 GB DDR3 @ 586MHz
Cache Size: 256 KB

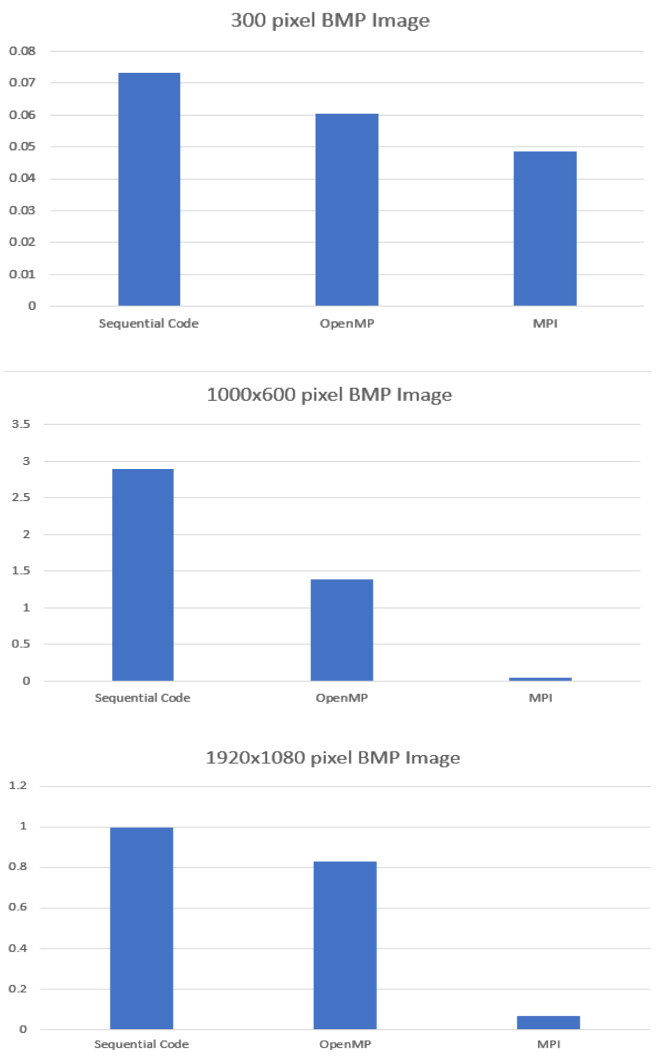

Figure 6: Speed-up observed in Sobel Edge Detection

For Gaussian Blurring, the calculations were done using:
Processor: AMD Ryzen 5 5600H
Memory: 16 GB DDR5 @3200 MHz
L1 Cache: 384 KB

The evaluation of the parallel execution performance is measured with respect to speed up with reference to the time taken for both sequential and parallel processing. Speedup measures how much a parallel algorithm is faster than a corresponding sequential algorithm. The hardware limitations of the system prevented us from achieving a high speedup, and hence the speedup achieved in our project was around 1.3 when OpenMP was used, which was much less than that obtained in the literature.

It is certainly evidence that parallel multicore Sobel and Gaussian blurring algorithms improves the performance of the traditional sequential Sobel algorithm by fully utilize the CPU to its utmost potential. This work is not limited to image processing methods only but can be extended to other processor intensive application.

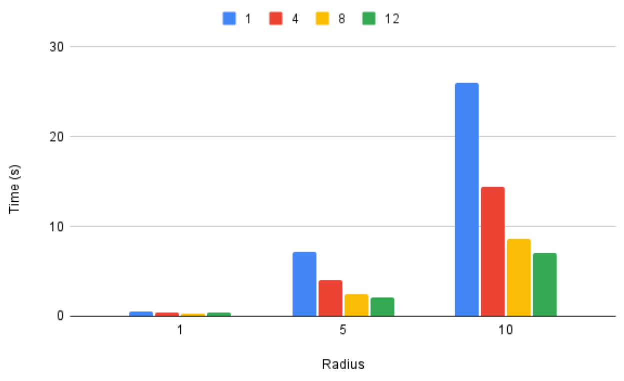

*Figure 7: MPI performance improvement for a 1920*1280 BMP image*

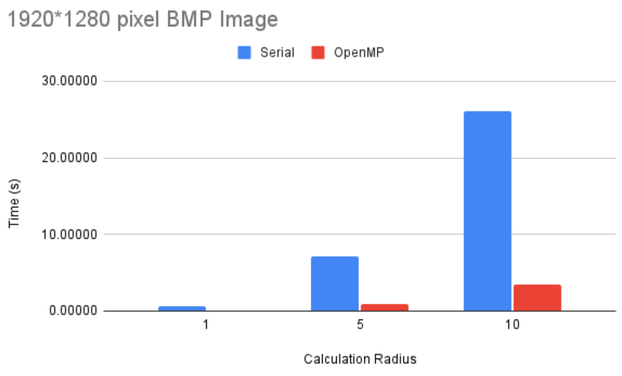

*Figure 8: OpenMP performance improvement for 1920*1080 BMP Image*

References

[1] ArpitaGopal, SonaliPatil, AmreshNikamInternational Journal of Computer Science and Application Issue 2010. A Parallel Algorithm for Image Edge Detection using difference chain encoding.

[2] Raman Maini, HimanshuAggarwalStudy and Comparison of Various Image Edge Detection Techniques. International Journal of Image Processing (IJIP), Volume (3) : Issue (1).

[3] F. M. Waltz and J. W. Miller, “Efficient algorithm for Gaussian blur using finite-state machines,” in Proc. SPIE Conf on Machine Vision Systems for Inspection and Metrology VII, vol. 3521, pp. 334-341, 1998

[4] Gonzalez, Rafael C. “Digital Image Processing”, Pearson Education, Inc., publishing as Prentice Hall, 2008.

[5] A. Kamalakannan and G. Rajamanickam, "High Performance Color Image Processing in Multicore CPU using MFC Multithreading," International Journal of Advanced Computer Science and Applications, Vol. 4, No. 12, 2013