ME-766 High Performance Scientific Computing

Parallelization of Image Processing algorithms in Python using PyMP

Introduction

Images are essentially matrices of real numbers describing the intensity value at those pixels. Hence, image processing algorithms are essentially calculations involving matrix operations that can be parallelized.

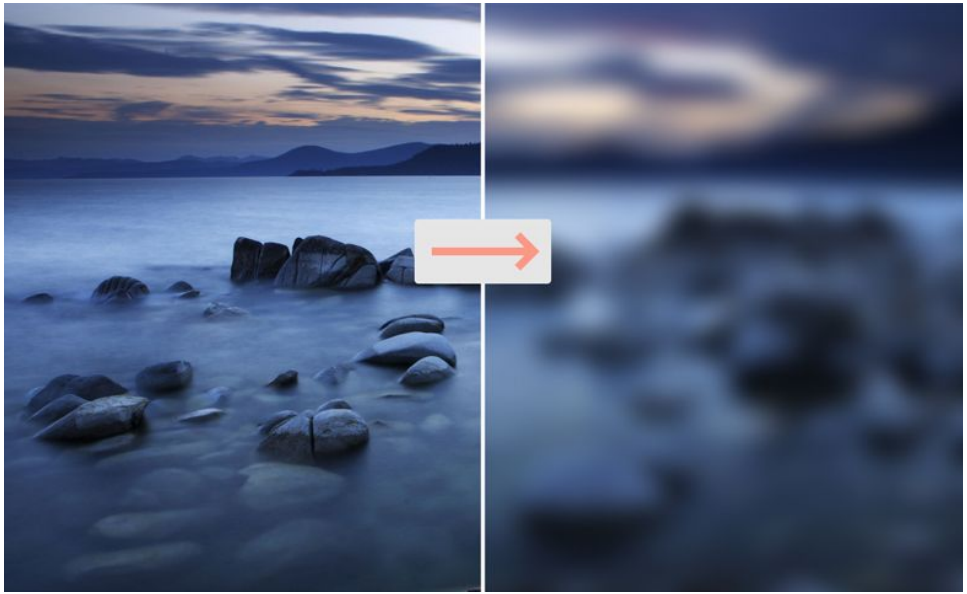
We use the PyMP package for python to achieve the parallelization. Pymp is based on OpenMP.

We implement two algorithms as a part of this project.
The Gaussian Blur and The Sobel Operation

The Gaussian Blur

This is a low pass filter for images, its most basic use is to remove high frequency noise present in the image.

The Gaussian Blur is essentially a 2d-convolution operation over the entire image. The 2d-convolution operation is performed between the image and a small matrix (we take 3x3 in our case).



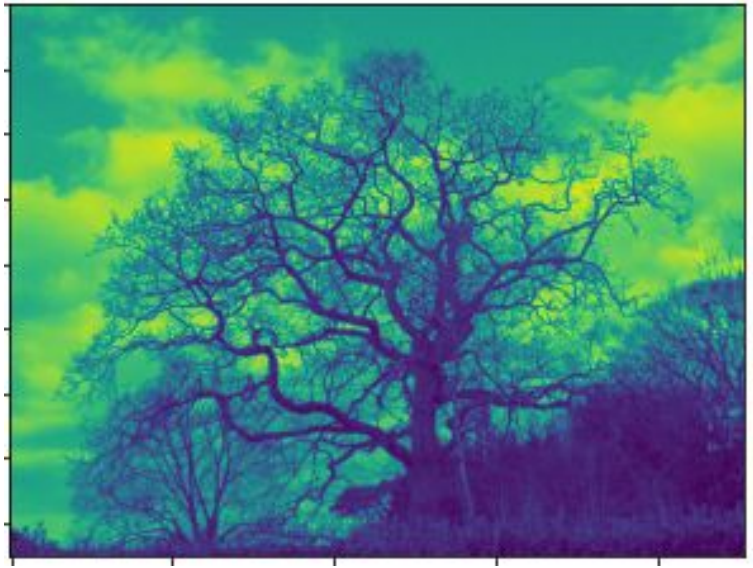
Applying Gaussian Blur to an Image

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Example of a convolution operator for performing the Gaussian Blur

Gaussian Blur was implemented in Python with the PyMP package for performing the parallelization task. The number of threads was set 4, and the program was run on a dual core processor.

Original Image



After applying Gaussian Blur

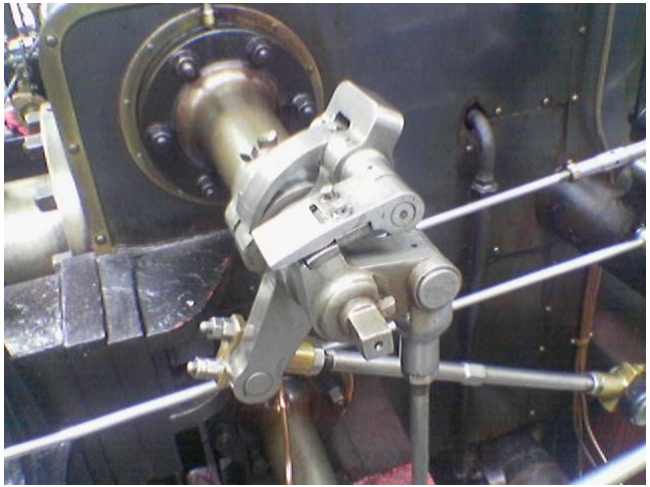
The Sobel Operation

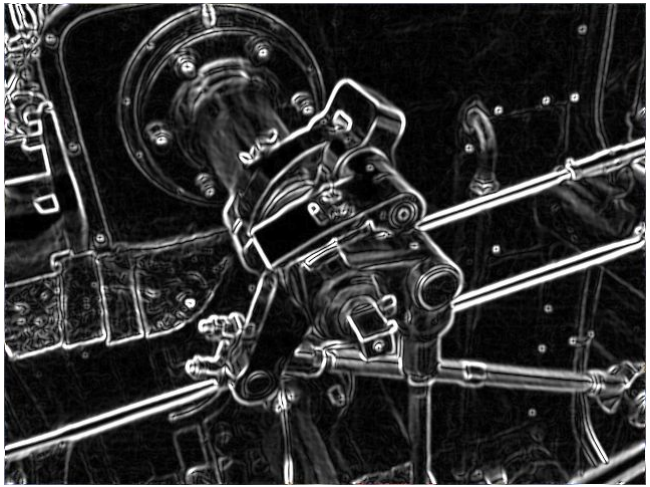
The Sobel operation is a popular way used for edge detection in images.

The Sobel operation typically carries out an image gradient operation. It is a discrete differentiation operation which is calculating the gradient of the image intensity function. After applying the Sobel Operation, each point represents the norm of the corresponding gradient vector at that point.

Two convolution operations are performed. The first convolution operation captures the gradient in the horizontal direction, and the other one in the vertical direction. Then they are squared and added pixel-wise to give the gradient norm at each pixel location.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$





$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

Example from Wikipedia

Sobel Operation was implemented in Python with the PyMP package for performing the parallelization task. Parallelization was done for the first convolution operation, the second convolution operation, as well as for summing the result of the two.

Original Image

After applying Sobel Operation

Timing analysis

The four cases were run on a dual core machine. The image was of size 2272x1704. The python package PyMP was used for parallelization with 4 threads. Machine used for running programs having 4 cores.
For the Sobel Operation case, each of the three steps were parallelized individually and the cumulative time is recorded.

	PyMP parallelization	Serialization
Gaussian Blur	11.75 s	40.61 s
Sobel Operation	27.36 s	90.92 s

As expected, the convolution operations are performed very fast when they are done in parallelization. And the results would scale as the size of image increases.

Contribution of each team member

- Rishi Agarwal: Implementing the Sobel Operation
- Manoj Bhadu: Implementing the Gaussian Blur
- Piyush Onkar: Timing Analysis, debugging
- Prince Sharma and Kaustubh Tendolkar: Presentation and report preparation, debugging

Thank you