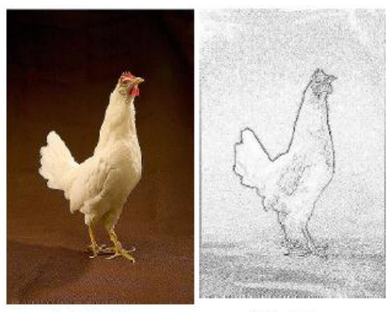
Parallelization of Sobel Edge Detection Algorithm

Haneen Tartouri
Supervisor: Mohammed Al-Dasht
Master of Informatics
College of Graduate Studies
Palestine Polytechnic University

Introduction

In this project , we proposing a design to parallelize the edge detection which is an active research filed and has used in many applications in different areas, especially in medical applications.
To detect the edge of image we need to apply specific computation on each pixel in the image, and that process will take a large time to do it especially for large size images, so in this project we propose a parallel algorithm for edge detection, in order to reduce the computation time of the edge detection operation.
So in this project we propose a new approach to parallelize the Sobel edge detection algorithm depending on domain and function decomposition and decentralization architecture of processors.



(a) Before (b) After

Figure 1: Image before and after edge detection

Parallel Computing Using Massage Passing Interface

In this project we have used the Message Passing Interface (MPI).
Also, in this project we depended on decentralization architecture for processors to achieve message passing approach, see Figure 2.

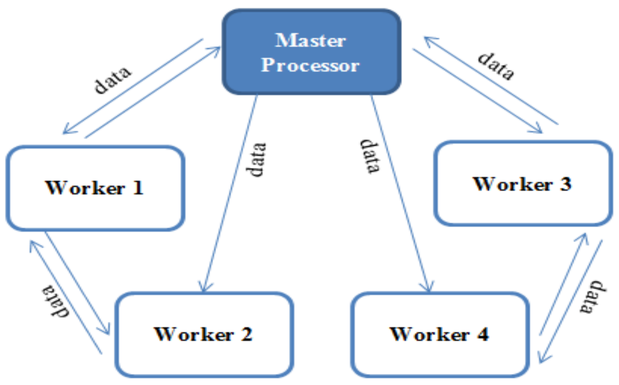


Figure 2: Decentralization architecture

Design and Implementation

1. Decomposition

In this project we have used two options of decomposition ; Domain and functional decomposition.
Figure 3 shows the domain decomposition.

Figure 3: Example of decomposition for the image where the number of worker processors equals 8

For the functional decomposition we decomposed the compute of gradient into two processor each processor compute the gradient in one direction.

2. Communication

In our method we have a local communication because our architecture is decentralized, and we don't need a global communication.

3. Synchronization

In this project we have a synchronization of communication depending only on send and receive message between the processor.

4. Load Balancing

In this paper we used the static load balancing, and we achieve it at two stages:

- First stage was done by equally partition the work between processor because we have the same power of processor, where each worker processor work on the same size of segment from image, and also the master work on some exceed rows of images to find the gradient.
- Second stage was done by equally distribution the segment after finding the gradient between the level 1 workers to find the magnitude in order to distribute the work of the master.

Experiment and Results

The experiment was implemented on one machine. This machine have 2 processors (Core 2 Due), 2GB RAM, and 2.1 GHz processor's speed. The software required to perform the parallel process are Ubuntu 11.10, lam-MPI library.
The experiment was done on 306x350 pixels, 512x512 pixels, 2560x1440 pixels, 4752x3168 pixels images.
The Parallel results are discussed based on speedup, performance improvements and efficiency.

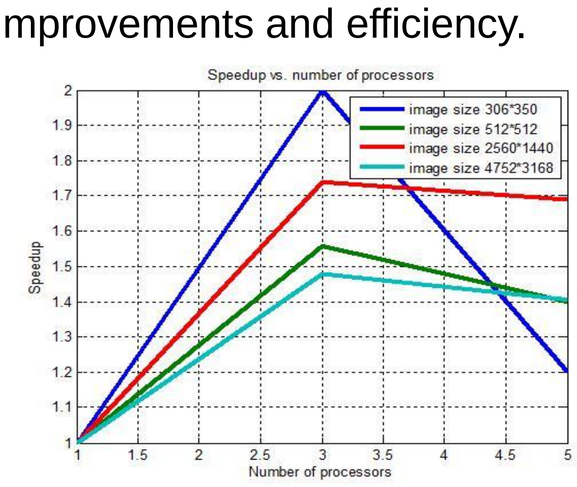


Figure 4: speedup for 4 different images size using different number of processors

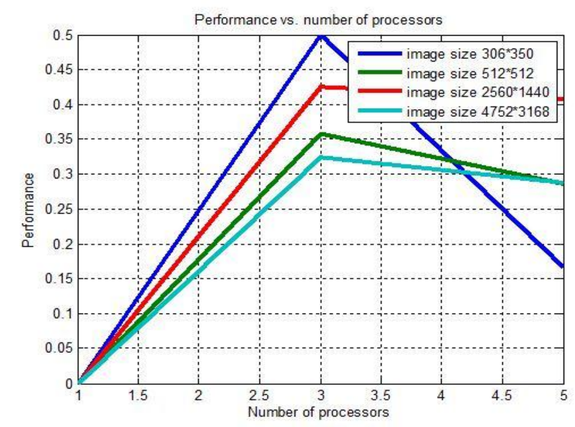


Figure 5: Performance improvement for 4 different images size using different number of processors

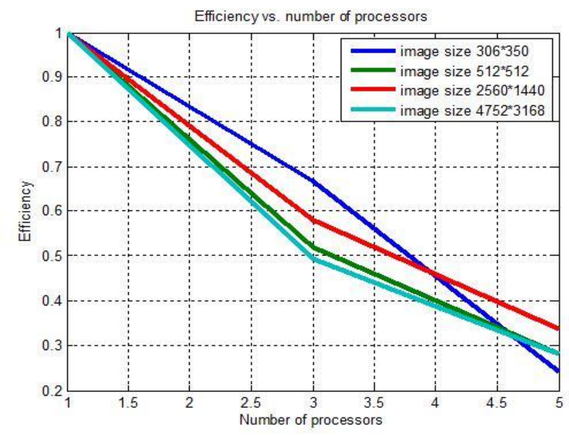


Figure 6: Efficiency for 4 different images size using different number of processors