Cmpe 300 Programming Project

"A parallel algorithm for image denoising with

the Ising model using Metropolis-Hastings algorithm"

Name: Serkan Özel

Submitted Person: Burak Suyunu

Due: 26.12.2018 - 23:59

Introduction:

Metropolis Hastings algorithm:

In statistics and statistical physics, the Metropolis–Hastings algorithm is a Markov chain Monte Carlo method for obtaining a sequence of random samples from a probability distribution from which direct sampling is difficult. This sequence can be used to approximate the distribution or to compute an integral.

Monte Carlo Method:

In computing, a Monte Carlo algorithm is a randomized algorithm whose output may be incorrect with a certain probability.

The Ising Model:

The Ising model, named after the physicist Ernst Ising, is a mathematical model of ferromagnetism in statistical mechanics. The model consists of discrete variables that represent magnetic dipolemoments of atomic spins that can be in one of two states (+1 or -1). The spins are arranged in a graph, usually a lattice, allowing each spin to interact with its neighbors.

So,

- The pixels of images will have two states (+1 or -1)
- We will be considering neighbours of a pixel

Ising model's meaning in this project:

If we assume that a black and white image is generated using The Ising model, then it means that if we take random black pixel from the image, it is more likely that this pixel is surrounded by black pixels (same for the white pixels).

Program Interface:

In order to start this program you need to install mpi. Version 1.4.4 recommended just for this project. After you installed the mpi you can compile main.cpp file with this command :

mpic++ main.cpp

Program Execution:

After compiling, run the program with this command:

mpirun -n -number of Processors- -name of the executable file generated-

-noisy input file path- -denoised output file path- -beta- -pi-

Here beta means how closely related the neighbour pixels are, according to The Ising Model. Between 0 and 1.

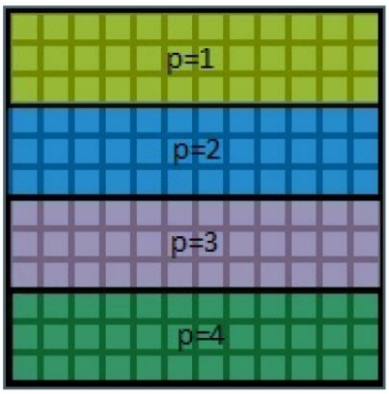
Pi means noise rate. If it is 0.1 this means %10 percent of

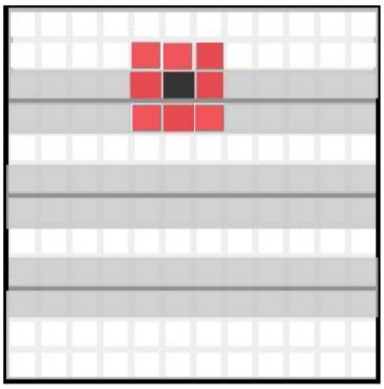
the pixels randomly toggled.

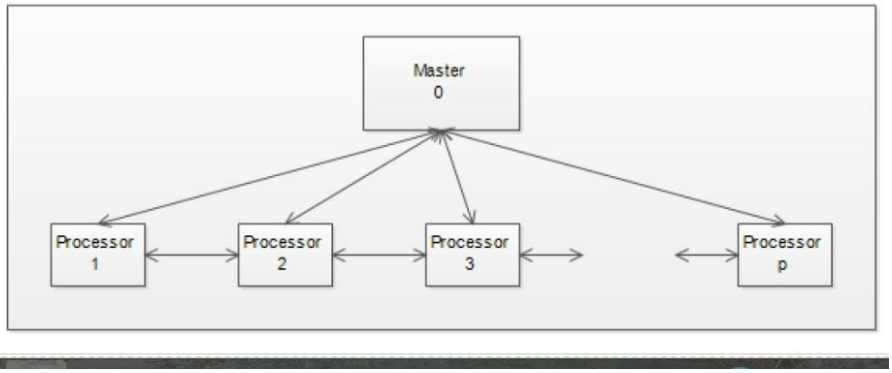
Input and Output

Input file consist of 200 lined text files, 1 or -1's seperated by a space. In each line there is 200 numbers. This file represent a noisy image. Output file is again a 200x200 similar file. It is denoised by the program.

Program Structure







In this project, each processor is responsible from a group of n/p adjacent rows. Each processor works on (n/p × n) pixels and these pixels are stored locally by the processor. When a pixel is inspected, the processor should inspect all of its neighbors. When the pixel is not on the boundary of two adjacent processors, information about the neighbor pixels are present to the processor.

When a pixel on the boundary is inspected, the processor needs to obtain information from the adjacent processor. Consider the black pixel in the below figure. Processor 1 should communicate with Processor 2 in order to learn the status of the south neighbor of the black pixel. Therefore, each processor should communicate with the adjacent processor at every iteration. Information about those neighbor pixels of the boundary can be stored locally and updated at every iteration.

Examples

Original Image:

Noisy Image:

Denoised Image :

Improvements and Extensions

In this program there are unnecessary data exchanges between processors. They can avoided by checking necessary moments.Also there may be a different processor-pixel allocation rather than horizontal divisions.According to processor architecture and pi values we may get better results.

Difficulties Encountered

Mpi environment is definitely hard to debug. Debugging can be done by assertions.

Conclusion

Was a good project thanks for giving this to us.