Digital Systems

EEE4084F

Practical 1: Pthread Implementation of Median Filter

Introduction

Pthreads is a C extension for Posix systems that is fairly easy to use for implementing function level parallelisation of code.

Median filters utilise basic statistics to remove extraneous values/noise from data sets. These data sets can be of an arbitrary number of dimensions, but for this prac you will only be working with a 2D data set, in particular JPG images.

For this prac you can work in groups of two or by yourself. In the report please clearly state your name and your partner's name (if applicable).

The prac starts by providing an easy Pthreads example. This is intended as code that you can build upon in order to deliver what is needed for this prac.

Pthreads Example

The folder "Sample Code Blur Filter" provides a complete working example of the radial blur function. The Sample Code Blur Filter folder contains:

Makefile: For compiling the program.

blur.c: The file that has the pthreads radial blur code.

serial.c: This file just has the file IO and timing code written for you.

To run the blur.c code use the following command

. /blur #(number of runs) #(number of threads) (input file name) (output file name)

Example, ten runs with four threads

. /blur 10 4 fly.jpg blur.jpg

Method of Implementing Median Filter

How to implement the median filter:

Firstly, make a 9x9 integer matrix - this will be your windowing matrix. Centre the windowing function on the pixel of the image that is to be replaced with the output of the median filter. Then copy of all the pixel contained in the area of the windowing matrix to the windowing matrix. If the windowing matrix goes out of the bounds of the image matrix (i.e. when your pixel is close to the edge of the image), use the 0 as the value for the out of bound pixels.

When the windowing matrix has been filled, it needs to be sorted. To sort the matrix, first sort each row individually, then sort the centre column which is the fifth column. Write the median value of the matrix, the 5th row of the 5th column, to the output file as the value for the pixel under consideration.

The code that you have been given does the file IO for you, because the images you will be using will be jpg, which utilise a non-trivial compression scheme. The data read in from the file will be in a 2D matrix with each pixel being a set of 3 unsigned char values (0 to 255), red, green and blue, and this is shown in table 1. Run the median filter on each colour on its own.

Table 1: Shows the layout of the data that you need to access the image data.

	0	1	2	3	4	5	6	7	8
0	R	G	B	R	G	B	R	G	B
1	R	G	B	R	G	B	R	G	B
2	R	G	B	R	G	B	R	G	B
3	R	G	B	R	G	B	R	G	B

In table 1, the boxed out area is just one pixies with is made up of a set of three unsigned integer values is the magnitude of their represented colour.

You have been provided with three pictures in a folder called Pic. Whilst you are testing your code just use fly.jpg because it the smallest of the images.

What You Need To Do:

For this prac you need to implement a median filter as described above. First implement a serial version and make sure it works, before moving to parallel code. There are three methods that you need to implement.

First is chunked, where each thread gets a chuck of rows equal to the total number of rows divided by the number of processors (so that each thread will run on a processor).

Second is dynamic interleaved, this is where each thread take the next available row and process it. Use a number of threads equal to the number of processors.

Third is dynamic chunks, this is similar to dynamic interleaved but instead of just doing one row at a time, do a few rows done before requesting a new set. For this section, the chunk sizes are 2, 4, 8, and 16 rows.

These can be done in the same program with the use #define commands.

Short Report

Write a short report (it can be one or two pages) discussing your observations and results. In the report I would like to see some of your images you produced, demonstrating your understanding of the purpose and method of the median filter.

I would also like to a break down of the different timing results you got from the different method. This data needs to be shown in a table, with the number of threads on the y-axis and the different processing types on the x-axis. Also, there needs to be a graph plotting the speed up factor observed versus the ideal speed up. The ideal speed up would have the speed up factor being perfectly proportional to the processing resource allocation. So if two processors are being used, then the ideal speed up factor would be 200%.

Comment on the results you got, and why you think you got them. Please reference technical factors in justifying your results.

Hand in procedure: Submit your report using the Prac01 VULA assignment.