High Performance Digital Embedded Systems
Practical 2 - Static pThreads

Jonathan Oehley[†] and Alan Pohl[‡]
EEE4120F Class of 2019
University of Cape Town
South Africa
[†]OHLJON001 [‡]PHLALA005

Abstract—**This report details an investigation into the use of multi-threaded applications in partitioning data for a median image filter. A golden measure, a serial implementation, was used to draw comparisons to an optimized and threaded implementation.**

I. Introduction

Things to still deal with:

- check language against Keegan's Prac1 feedback
- add image sizing

A Median Filter is a filtering method often used in image and signal processing. On a 2-dimensional image problem, each colour component of the output pixel is the median of the surrounding n x n pixel colour components. Such a filter is often implemented to reduce speckle or salt-and-pepper noise in images [1].

This investigation will implement a 9x9 median filter.

II. Methodology

The implementation of a median filter requires some definition around the pixels near the edge of an image. The outer edge-case pixels were calculated by the following means; the area over which the pixel component values were compared with to determine the median was simply truncated by the edge of the image. In other words, for the pixels in the furthest corners, the area over which the median was determined was 5 x 5 pixels.

A. Golden Measure

The golden measure was created as a relatively simple block of code that's main objective was to be a working model. From this, further implementation could be compared and a comparison drawn.

The golden measure sorting method was implemented as a simple bubble sort. While not the quickest sorting technique, the bubble sort is easy to code and understand. Therefore, it was possible to implement the sorting method relatively quickly in this manner. The filter was simply implemented by flattening the 2 dimensional comparison area, into a single array before passing it to the sorting algorithm to determine the median.

B. Multi-thread Implementation

This implementation looked to improve the execution time of the golden measure. This was done using two techniques.

Firstly, instead of implementing a bubble sort by default, a investigation was made between bubble sort, a select sort function (which only sorted just above halfway to get the median) and the hybrid sort function (std::sort()) built into c++. The fastest sorting method was used in this implementation which will be discussed in the Results section.

Secondly, the golden measure implementation was converted into a multi-threaded application. This was done through data-partitioning of the image into rows. This allows the data sorting tasks to be split up over multiple processors to decrease the overall execution time of the application

C. Experiment Procedure

The program was coded using a shared git repository to allow collaboration between the two students. The code was tested periodically to ensure functionality of each function.

Testing was conducted by running the program a few times to ensure that it was stored in cache before taking readings. Then the program was run with different inputs, number of threads and sorting functions. The execution time for each case was measured for multiple runs of the program and averaged across them. This generates a single value for the execution time of the program with those parameters.

III. Results

The results section is for presenting and discussing your findings. You can split it into subsections if the experiment has multiple sections or stages.

A. Figures

Include good quality graphs (see Fig. 2). These were produced by the Octave code presented in listings 1 and 2. You can play around with the `PaperSize` and `PaperPosition` variables to change the aspect ratio. An easy way to obtain more space on a paper is to use wide, flat figures, such as Fig. 3.

Always remember to include axes text, units and a meaningful caption in your graphs. When typing units, a µ sign has a tail! The letter "u" is not a valid unit prefix. When typing resistor values, use the Ω symbol.

TABLE I

SORTING ALGORITHM COMPARISON

Image	Bubble	Select	Qsort
Fly	25.23	7.75	4.66
Alan	9.43	3.76	1.95
Average	17.33	5.75	3.30
Speed-up	1.00	3.01	5.25

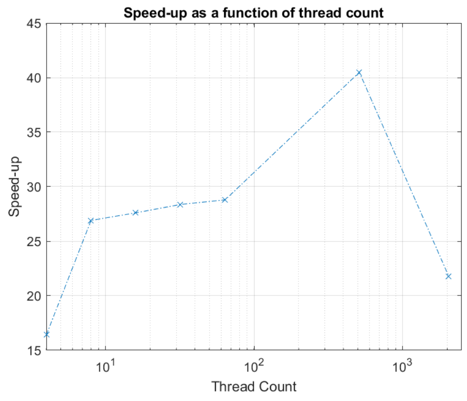

Fig. 1. Speed plz

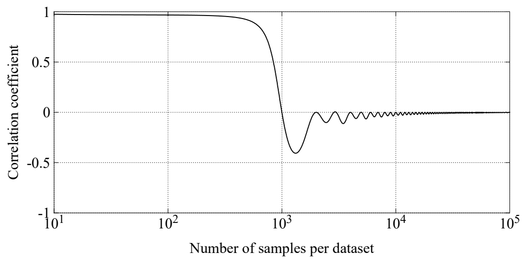

Fig. 2. The correlation coefficient as a function of sample count.

```
function FormatFig(X, Y, File);
 set(gcf, 'PaperUnits'      , 'inches');
 set(gcf, 'PaperOrientation', 'landscape');
 set(gcf, 'PaperSize'       ,       [8, 4]);
 set(gcf, 'PaperPosition'   , [0, 0, 8, 4]);

 set(gca, 'FontName', 'Times New Roman');
 set(gca, 'Position', [0.1 0.2 0.85 0.75]);

 xlabel(["\n" X]);
 ylabel([Y "\n\n"]);

 setenv("GSC", "GSC"); # Eliminates stupid warning
 print(...
  [File '.pdf'],...
  '-dpdf'...
 );
end
```

Listing 1. Octave function to format a figure and save it to a high quality PDF graph

B. Tables

Tables are often a convenient means by which to specify lists of parameters. An example table is presented in table II.

```
figure;                                   # Create a new figure
# Some code to calculate the various variables to plot...
plot(N, r, 'k', 'linewidth', 4); grid on; # Plot the data
xlim([0 360]);                            # Limit the x range
ylim([-1 1]);                             # Limit the y range
set(gca, 'xtick', [0 90 180 270 360]);    # Set the x labels

FormatFig(...                             # Call the function with:
 'Phase shift [\circ]',...                      # The x title
 'Correlation coefficient',...                  # The y title
 ['r_vs_N;_f=' num2str(f) ';_P=' num2str(P)]... # Format the file name
);
close all;                                # Close all open figures
```

Listing 2. Example of how to use the FormatFig function

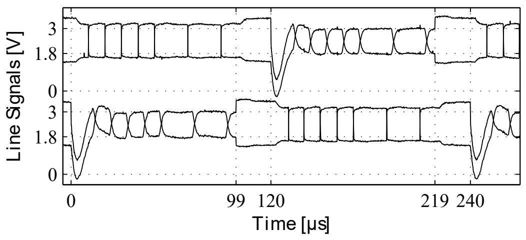

Fig. 3. Oscilloscope measurement showing physical line signals on both ends of a transmission line during master switch-over [2].

TABLE II

MY INFORMATIVE TABLE

Heading 1	Heading 2	Heading 3
Data	123	321
Data	456	654
Data	789	987

C. Pictures and Screen-shots

When you include screen-shots, pdfLaTeX supports JPG and PNG file formats. PNG is preferred for screen-shots, as it is a loss-less format. JPG is preferred for photos, as it results in a smaller file size. It's generally a good idea to resize photos (not screen-shots) to be no more that 300 dpi, in order to reduce file size. For 2-column article format papers, this translates to a maximum width of 1024. **Never change the aspect ratio of screen-shots and pictures!**

D. Maths

LaTeX has a very sophisticated maths rendering engine, as illustrated by equation 1. When talking about approximate answers, never use ± 54 V, as this implies “positive or negative 54 V”. Use ≈ 54 V or ~ 54 V instead.

$$y = \int_0^\infty e^{x^2} \mathrm{d}x \tag{1}$$

IV. CONCLUSION

The conclusion should provide a summary of your findings. Many people only read the introduction and conclusion of a paper. They sometimes scan the tables and figures. If the conclusion hints at interesting findings, only then will they bother to read the whole paper.

You can also include work that you intend to do in future, such as ideas for further improvements, or to make the solution more accessible to the general user-base, etc.

Publishers often charge "overlength article charges" [3], so keep within the page limit. In EEE4084F we will simulate overlength fees by means of a mark reduction at 10% per page. Late submissions will be charged at 10% per day, or part thereof.

References

[1] G. R. Arce, *Nonlinear Signal Processing: A Statistical Approach.* John Wiley and Sons, 2005.

[2] J. Taylor and J. G. Hoole, "Robust Protocol for Sending Synchronisation Pulse and RS-232 Communication over Single Low Quality Twisted Pair Cable," in *Proceeding of ICIT*. Taiwan: IEEE, Mar. 2016.

[3] "Voluntary Page and Overlength Article Charges," http://www.ieee.org/advertisement/2012vpcopc.pdf, 2014.