Competition

ECE408 Applied Parallel Programming Project/Competition

Date Due: December 14, 2012

University of Illinois at Urbana-Champaign

Daniel Rasinski

Manny Lau

Tommy Shiou

Table of Contents

[Introduction 2](#_Toc343541987)

[Problem Description 2](#_Toc343541988)

[Implementation 2](#_Toc343541989)

[Results 3](#_Toc343541990)

[Challenges 3](#_Toc343541991)

[Future Work 4](#_Toc343541992)

[Conclusion 4](#_Toc343541993)

[References 4](#_Toc343541994)

# Introduction

The ScanCath System is a C-Arm based low-dose, real-time x-ray system intended for Cardiac Electrophysiology and Cath Lab imaging applications. This system has uses a different geometry than the conventional fluoroscope. A large scanning x-ray beam is emitted through collimator holes and directed to the detector to collect the image. The x-ray detector is illuminated at least once per collimator hole, which generates many images. These images are combined through a process called image reconstruction and the image reconstructed represents one focal plane. The x-ray emits beams multiple times because many focal planes are needed to clearly show different levels of the 3-D object. Many sequential algorithms are also implemented to produce better results.

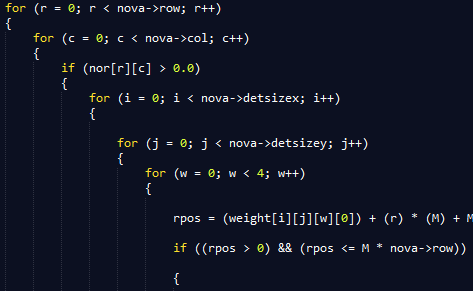
# Problem Description

The ScanCath system is slow because it requires complex algorithms which contain many nested FOR loops to perform its image reconstruction. First, the dynamic range equalization algorithm containing 3 quad-nested FOR loops are called to remove bad pixels. Then the code loops through the number of planes and while calling two time-consuming functions and more loops along the way. These two time-consuming functions contain 18 coalesce FOR loops. Because of the outer FOR loop that calls these functions, the time these functions take to process are multiplied by the number of planes (96) which produces significant time consuming code.

# Implementation

The overall strategy on improving performance is to seek and break down the FOR loops into multiple chunks and have many threads work on the function in parallel.

The first functions targeted were the functions inside the outer FOR loop that went through the planes. One of these functions was the reconstruct function. It had the biggest impact on performance because of its penta-nested FOR loops and its call to another function: Edge Gain.



The first two FOR loops of the penta-nested FOR loops were parallelized. So we originally called the kernel using the x and y dimensions to break down r and c, and further added w to its z dimension.

In order to successfully use the kernel for computing in parallel, we changed all the 2d-4d arrays into 1d. After peeking into initial edge gain function, we saw that the array (\*\*normin) it modified looped through the exact same parameters. So we put this array in the same area to avoid redundancy which improves performance.

The following grid/block dimensions and kernel declarations:



\*Reconim and \*normin were cudaMalloc’ed on the device because these were the arrays updated by the reconstruct and initialize edge gain functions. The edge gain function that followed the initialize edge gain function also modifies and uses allocated reconim\_d and normin\_d. Unfortunately, this function used a greater number of column and row, so we were unable to combine these two functions into one kernel call. We invoked another kernel call to remove the FOR loops in edge gain, reusing the allocated device memory reconim\_d and normin\_d to reduce the amount of cudaMemcopies (takes time) used. A new grid was used to accommodate the new row and col dimensions:



The second function we parallelized was initWeight. We parallelized the 4 FOR loops by invoking the kernel with the correct dimension parameters. This method was very similar to the parallelization of the reconstruct functions.

The dimensions of the rows and columns in these functions were either 160x80 or 1600x800. Either or, the biggest tile size we were able to use that was divides 160x80 with no divergence was 16.

Another strategy was to remove unnecessary code that affected performance.

We also consolidate combinable code.

# Results

[Present and discuss your results]

# Challenges

[Discuss the main challenges you faced while tackling the problem. You may find yourself repeating a few things you already discussed in the Implementation section. That’s okay, but this section is where you should discuss the challenges at length.]

# Future Work

[Discuss any idea you would have liked to try but did not have the time to do so.]

# Conclusion

[Conclusion]

# References

Lowell, Augustus. Design Specification ScanCath Reconstruction Algorithms. Triple Ring Technologies.