Final Project Proposal
Autumn 2024 - EE P 524: GPU Computing
Van Tha Bik Lian
Sai Jayanth Kalisi
November 12, 2024

## 1. Project Overview

This project will focus on optimizing 3D stencil-based kernels for edge detection on GPUs - Option 3 in the list of Project focus options provided. Edge detection in 3D volumes is crucial in fields like medical imaging, where detecting boundaries within volumetric data aids in visualizing structures such as organs or tissues. By implementing and optimizing 3D stencil operations for edge detection on a GPU, this project aims to improve processing speeds and memory efficiency for high-resolution 3D datasets. Using tools in this class, like Nsight Compute, this project aims to enhance overall kernel performance, ***particularly for 3D stencil patterns,*** taking inspiration from the class' 2D stencil approach***.***

---

## 2. Technical Summary

- **Problem Scope**: The project will implement 3D versions of the Sobel or Prewitt operators - those predominantly used in edge detection. This approach will calculate the gradient of intensity at each 3D pixel by referencing its neighboring 3D pixels - a staple requirement for stencil kernels.
    - Given the 3D structure, each 3D pixel, or voxel as is normally referred to, requires accessing a 3x3x3 or larger grid of surrounding voxels, creating more complex memory access patterns compared to 2D edge detection.
    - We will look at optimizing this stencil grid from 3x3x3 to larger values, like 5x5x5.
- **Key Computations**: In 3D edge detection, each voxel in the volume is updated based on the weighted sum of its neighboring voxels across three axes (x, y, and z). ***This increases the arithmetic intensity and memory traffic****,* particularly for high-resolution 3D images. Managing these computations efficiently is crucial, as the number of memory accesses and control divergence both increase significantly with 3D data.
    - Overall, a 2D kernel is quite similar to a 3D kernel in how things are set up. The control divergence will still be based on the excess size generated by the nature of convolution itself. To remedy this, we will look into **padding the base 3D image,** and see how that could potentially mitigate further divergence.
- **Parallelism Identification**: Each voxel's gradient magnitude ***can be computed independently,*** making it feasible to parallelize across GPU threads. However, careful memory management is essential to reduce global memory accesses, as the 3D stencil pattern requires multiple neighbors along each axis, intensifying memory demands.
    - Shared memory is crucial. We will be putting the stencil in shared memory to reduce data transfer. *Tiled base matrix might be something to look into as well.*

---

**3. Optimization Strategy**

The optimization matrix will include tuning configurations for 3D kernel parameters such as:

- **Grid/Block Launch Configurations/Tile Sizes**
- **Shared Memory Usage**: for both stencil and base matrix
- **Base Operator type**: Sobel, Prewitt and other edge detection kernels will be looked at and compared. Simple change in the filter should be enough.
- **Boundary Handling**: Efficiently managing boundary cases where complete 3D neighborhoods are not available, reducing control divergence due to edge conditions.
    - *Test the effectiveness of reducing control divergence with padding.*
- **Coarsening Factor**: Adjusting the number of voxels each thread processes to increase arithmetic intensity. Due to complexity of implementation for this, we will consider this as a possible additional thing to look at. **Not a part of our core deliverable (REACH).**

For this 3D implementation, the effect of stencil size (e.g., 3x3x3, 5x5x5, or larger) on both accuracy and throughput metrics will be evaluated

For our deliverables, this would mean that there are 5, potentially 6, kernels that we would design and implement. These include:

- CPU “Kernel” for edge detection to act as baseline
- GPU Global memory Kernel given in Class
- GPU Kernel where Filter is in Shared memory
- GPU Kernel where Filter and dataset is in shared memory (Tiled)
- GPU Kernel with no handling of padded divergence
- (REACH) GPU Kernel with coarsening factor of 2 voxels per thread, or more.

Our Output EXEs will look like:

- **Tile size** variation for tiling only: 4x4x4, 8x8x8, *8x8x16(TBD)*
- **Kernel size** variation for all three types: 3x3x3, 5x5x5
- **Base Operator** modification for all three types: Sobel, Prewitt, Canny (TBD), Robinson (TBD)
- **Boundary Handling** for tiling only: Padding, no Padding
- **Kernel Type**: Global, Stencil_Only, Stencil_and_Image
- **Coarsening Factor** for Tiling only (REACH): 1 and 2.

---

**4. Role of CUDA and Parallel Patterns**

CUDA will sanction efficient implementation of 3D edge detection filters on the GPU:

- **Constant Memory**: Storing the convolution filters for edge detection in constant memory, accessible to all threads without modification.

- **Shared Memory Tiling**: Implementing a 3D shared memory tiling strategy, where blocks load 3D voxel tiles into shared memory for local computations, reducing global memory accesses.

This design aims to improve the arithmetic-to-memory access ratio and optimize memory usage for large 3D volumes.

---

**5. Testing and Validation**

NSight Compute provides Speed of light, Compute, Memory, Occupancy, and roofline plots. We intend to look at these values in particular for memory and throughput overview.

Latency metrics and active warps, as well as efficient use of the memory hierarchy systems will be looked into. Further metrics like L1 and L2 cache hit rates, bank conflicts and Sectors/request will be looked at for memory utility.

A basic table has been created here: https://docs.google.com/spreadsheets/d/1nPIY7L3BHG6XiCasgFNKrFSTcV0i-WUeaGxuU5geas/edit?usp=sharing

---

**6. References and Data Sources**

- **Primary References**: Course materials on convolution and stencil patterns, including Lecture 6 notes, Textbook on convolution(AUT24_GPU-Compute_Lecture)(ch7_ch8).
    - https://en.wikipedia.org/wiki/Sobel_operator
- **Datasets**
    - https://web.cs.ucdavis.edu/~okreylos/PhDStudies/Spring2000/ECS277/DataSets.html

---

**7. Expected Outcomes and Challenges**

The expected outcome is multiple high-performance 3D edge detection kernels, optimized for memory efficiency and speed on large 3D datasets. Anticipated challenges include managing the increased memory access in 3D stencils and minimizing control divergence at volume boundaries.