Assignment 4 - 1: CUDA Chamfer Distance

Introduction to Program Analysis and Optimization

Deadline: April 7, 2025

March 31, 2025

1 Overview

The Chamfer distance [2] is a metric used to measure the similarity between two point clouds or shapes. It has significant applications in computer vision, particularly in tasks such as point cloud registration, shape matching, and 3D object recognition. The Chamfer distance between two point sets A and B is computed by finding, for each point in set A, the nearest point in set B, and vice versa, then summing or averaging these distances.

Chamfer Distance [1] between two sets A and B is calculated as:

$$D(A, B) = \frac{1}{|A|} \sum_{a \in A} \min_{b \in B} \|a - b\|^2 + \frac{1}{|B|} \sum_{b \in B} \min_{a \in A} \|a - b\|^2$$

where

- A and B are two point clouds.
- a and b represent points in the sets.
- $\|a - b\|^2$ is the squared Euclidean distance between points a and b.
- The Chamfer Distance includes two components: one summing over the nearest points in B for each point in A, and the other summing over the nearest points in A for each point in B.

The goal of this assignment is to implement a CUDA-based solution for calculating the Chamfer Distance between two 3D point clouds. The Chamfer Distance is a metric used to measure the similarity between two point sets by evaluating the nearest point distances between them.

2 Deliverables

The structure of the template directory for this assignment is shown in Figure 1.

1. `Source Code:`
   - A CUDA-based implementation for calculating the Chamfer Distance between two point clouds.
   - Your CUDA kernel should efficiently compute the Chamfer Distance using parallelism.
2. `Input`:
   - Point clouds A and B, provided in CSV format, where each row corresponds to a point with 3D coordinates.

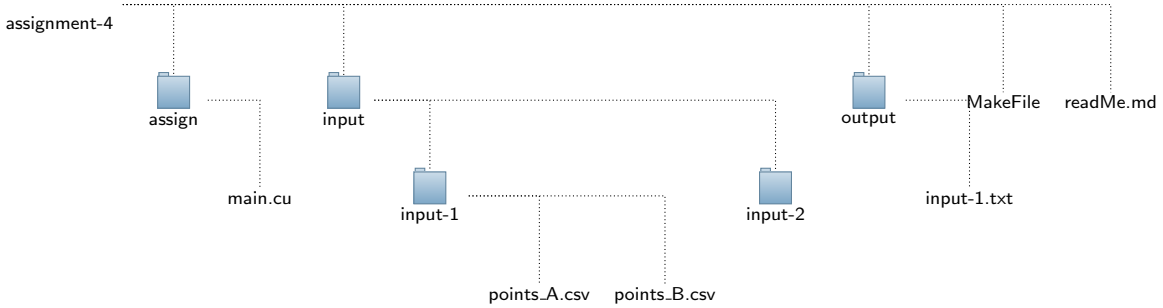

Figure 1: Directory Structure

- Sample datasets are provided in the "input" directory.

3. `Output`: The calculated Chamfer Distance value as a scalar output is stored in a result file.
4. `MakeFile`:
   - A Makefile that includes all necessary commands to compile and run your CUDA program.
   - The Makefile should allow the TAs to compile and run the code without any further modifications.
5. `readMe.md` This file should explain the process for compiling, running, and using your Chamfer algorithm on any dataset. The documentation should clearly describe:
   - The GPU you have used and its description (DRAM size, Architecture name etc.), the CPU you have used and its description.
   - How to compile the code.
   - How to run the CUDA program with input files.
   - Explanation of any input parameters or command-line arguments.
   - Performance analysis comparing the CUDA-based implementation with a baseline CUDA version.

   The instructions should be generic so that TAs do not need to modify the implementation to run it on different datasets.

3 Input

The input consists of multiple subfolders in the input directory, each containing two CSV files:

1. Point Cloud A (`points_A.csv`): Each row represents a point in A with n-dimensional coordinates, in the format: x, y, z
2. Point Cloud B (`points_B.csv`): Similarly, each row represents a point in B with 3D coordinates.

The datasets will be loaded into the program, and the Chamfer Distance between the point clouds will be calculated.

4 Output

The output of your program should be:

1. **Chamfer Distance Value:** A single floating-point number representing the calculated Chamfer Distance between the two point clouds.

2. **Elapsed Time:** You will time the code to determine (and print) the elapsed time (t1 - t0) in milliseconds to compute the chamfer distance (i.e. t0 = immediately after the CSV file is read ; t1 = just before printing the o/p)

3. **Output File:** The output file for each dataset should have the same name as the input subfolder, with the file extension `.txt` (e.g., `folder_1.txt` for the dataset in `folder_1`).

5 Performance Considerations

- You are expected to compare your CUDA-based implementation with a baseline CUDA program of the Chamfer Distance calculation.
- The efficiency of your CUDA implementation will be evaluated based on the time taken for calculations with varying sizes of input data.

6 Evaluation

Your implementation will be graded on the following:

- **Correctness**: Whether the calculated Chamfer Distance is accurate for provided test cases. The baseline python script used to compute the chamfer distance is shared for reference. Note: The normalization on the distances and also the square root of the sum of $diffs^2$ is used per Euclidean Distance metric definition (and not the direct sum of $diffs^2$).
- **Performance**: The efficiency of the CUDA implementation compared to the baseline version[1]. The extra points will be awarded if the CUDA implementation beats the baseline program in terms of execution time.
- **Code Quality**: Clean, readable, and well-documented code.
- **Documentation**: Completeness and clarity of the README file, including instructions on how to run and compile the program.

7 Submission Instructions

- Submit your code, the Makefile, the readMe.md, and the generated output files.
- Ensure that your code compiles and runs correctly by following the instructions in the README file.
- Ensure the output file matches the format specified above.

Note: Late submissions will be penalized 20% per day past the deadline.

References

[1] PARKCHEOLHEE-LAB. Chamfer distance. https://parkcheolhee-lab.github.io/chamfer-distance/, 2025. Accessed: Mar 27, 2025.

[2] SIMSANGCHEOL. Chamfer distance. https://medium.com/@sim30217/chamfer-distance-4207955e8612l, 2025. Accessed: Mar 27, 2025.

[1] Baseline code has not been provided in the assignment directory but will be provided separately later.