Presented to the College of Computer Studies

De La Salle University - Manila

Term 3, A.Y. 2023-2024

In partial fulfillment of the course

In CEPARCO S11

Data-level Parallelism Integrating Project Proposal:

MOVEMENT RECOGNITION IN SIMT

Group No. 5

Submitted by:

Cai, Edison B.

Dequico, Beverly Joyce P.

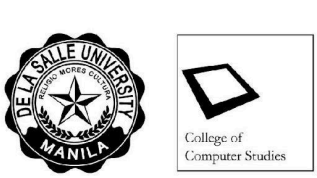
La'O, Erin Denise C.

Relucio, Jan Jhezaree L.

Submitted to:

Prof. Roger Luis Uy

June 21, 2024



Abstract

Modern devices, like mobile phones and smartwatches, are equipped with accelerometers that can easily record and provide access to accelerometer data. This data includes X, Y, and Z coordinates to describe the movement of a subject. By comparing these coordinates, trends, and inferences may be formed based on abrupt or gradual changes. The proposed project aims to create a system that recognizes movement trends from a given accelerometer dataset using Single Instruction, Multiple Threads (SIMT) in Google Collab with CUDA, enabling efficient and scalable data processing.

I. Description

a. Inputs

There will be three inputs:

- **Main Dataset Array**: This consists of arrays containing 3 *double* values each, representing the X, Y, and Z coordinates.
- **Integer N**: This denotes the number of coordinate sets to be grouped together for trend analysis.
- **Integer USER_FILTER:** This number corresponds to a specific trend type (e.g., linear, accelerating, rotational) that the user wants to filter in the program output instead of displaying all trends by default.

b. Proposed Process

i. Parallelized Processes

The primary focus is to leverage data-level parallelism for efficient movement recognition. The proposed process involves breaking down accelerometer data into manageable chunks that can be processed simultaneously using SIMT structure. The steps are as follows:

a. **Data Segmentation/Preprocessing:** The dataset will be broken down into smaller subsets, each containing N sets of coordinates.
b. **Trend Analysis for Each Segment:** Each subset of data will be processed by a separate CUDA thread, where trends such as linear movement, acceleration, or rotation will be detected through parallel computation.

ii. Existing Implementations

Movement Recognition using accelerometer data has been widely explored in various applications:

a. **Mobile Applications:** Fitness and health tracking apps like Google Fit and Apple Health use accelerometer data to monitor physical activities. In 2022, Brew, Faux, and Blanchard wrote a paper to check the effectiveness of a smartwatch app as modern technology can be used as an accessibility tools, especially for people such as the elderly, who may experience falls as their health deteriorates. They stated that falls were more likely to be detected if the smartwatch was on the same side as the fall, which implies that the point of reference for the movement data is an important factor to consider. There was also an observation that certain smartwatches and operating systems had better sensitivity.
b. **Academic Research:** Studies often utilize accelerometer data for diverse applications, from monitoring patients' rehabilitation to analyzing athletes' performance. For instance, in the study titled "Accelerometer Data: Applications in Rehabilitation and Sports Performance" by De Fazio et al., these data allow for defining the correct therapies and evaluating their effectiveness and patient progress based on suitable protocols that ensure the patients carry out their physiotherapy programs even at home. Moreover, they facilitate optimizing exercises and assessing athletes' progress during sports training.
c. **Machine Learning Systems:** Advanced applications use machine learning models to classify movement patterns from accelerometer data, providing detailed activity insights. For example, a study by Mauldin, Canby, Metsis, Ngu, and Rivera (2018) involved utilizing deep learning to create a fall detection model using a dataset obtained from an IoT device with its own accelerometer. The researchers mentioned that a deep learning approach proved to be

more accurate when compared to algorithms such as Support Vector Machine and Naive Bayes.

iii. How to Parallelize Proposed Process

The CUDA programming model will be utilized extensively to optimize performance through parallelism. First, leveraging CUDA's SIMT (Single Instruction, Multiple Threads) architecture is essential. This model allows multiple threads to execute the same instruction concurrently on different data points, enabling efficient accelerometer data processing across NVIDIA GPUs. Secondly, data chunking and parallel processing will be implemented to divide the dataset into manageable segments. Each segment will be processed independently by CUDA threads, ensuring parallel computation of movement trends. Thirdly, shared memory utilization will be prioritized to enhance data access speed and reduce latency. CUDA threads can efficiently access and manipulate data by storing frequently accessed data in shared memory, improving overall computational efficiency.

iv. Other Processes

a. **Aggregation of Results:** Results will be combined from each thread to form a comprehensive summary of the trend analysis.
b. **Output Filtering:** The system will simply print the result statements only for the relevant trend(s) depending on the user's input for USER_FILTER.

c. Outputs

The primary output will be a set of print statements indicating the movement trends for each subset of coordinates. The user can specify if only

subsets of a certain trend will be printed. These statements will detail whether the movement is linear, accelerating, or rotational.

II. Proposed Target Implementation Platform

The project will be implemented on a platform that supports GPU computing and CUDA for efficient parallel processing such as Google Collab, which offers free access to GPU resources and is suitable for collaborative coding and testing. CUDA will be used to develop the parallel processing algorithm that will run on Google Collab's T4 GPU, enabling the execution of multiple threads simultaneously. Using Google Collab also ensures that the code can run even if the user's computer does not possess an NVIDIA graphics card.

III. References

Brew, B., Faux, S. G., & Blanchard, E. (2022). Effectiveness of a Smartwatch App in Detecting Induced Falls: Observational Study. *JMIR formative research*, *6*(3), e30121. https://doi.org/10.2196/30121

Mauldin, Taylor & Canby, Marc & Metsis, Vangelis & Ngu, Anne & Rivera, Coralys. (2018). SmartFall: A Smartwatch-Based Fall Detection System Using Deep Learning. Sensors. 18. 3363. 10.3390/s18103363.

De Fazio, R., Mastronardi, V. M., De Vittorio, M., & Visconti, P. (2023, February 7). *Wearable Sensors and Smart Devices to Monitor Rehabilitation Parameters and Sports Performance: An Overview*. NCBI. Retrieved June 20, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9965388/