УДК 004.272

ПРИКЛАДНІ АСПЕКТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ НА ОСНОВІ ПАРАЛЕЛЬНО-ІЄРАРХІЧНИХ МЕРЕЖ

Матейчук Максим, Яровий Андрій

Вінницький національний технічний університет

Анотація

В роботі показано способи організації процесу паралельно-ієрархічної обробки інформації в CPU- та GPU-системах. Результатом є підвищення швидкодії розпізнавання зображень профілю лазерного променя на основі паралельно-ієрархічних мереж. Наведено алгоритми розв'язання поставленої задачі на основі CPU та GPU-орієнтованої апаратної платформи.

Abstract

It is shown how to organize the process of parallel-hierarchical information processing in CPU- and GPU-systems. The result is an increase in laser beam images recognition process based on parallel-hierarchical networks. The algorithms of solving the formulated problem based on GPU and CPU hardware platforms are given.

Вступ

Дослідження присвячене розпізнаванню зображень профілю лазерного променя на основі паралельно-ієрархічних мереж [1]. Під розпізнаванням розуміється визначення ступеня спотворення у відношенні до еталонного зображення та подальша класифікація динамічно змінюваних зображень. Вхідними даними системи є цифрові зображення профілю лазерного променя у кольоровій моделі Grayscale.

Метою дослідження є збільшення швидкодії оброблення зображень шляхом організації процесу паралельно-ієрархічної обробки інформації. Для цього було обрано високопродуктивний обчислювальний комплекс.

Задачами дослідження є: розробка і реалізація CPU-версії алгоритму, розробка і реалізація GPU-версії алгоритму розпізнавання зображень профілю лазерного променя на основі паралельно-ієрархічних мереж, порівняння результатів.

Реалізація на основі CPU-орієнтованої апаратної платформи

Алгоритм виконання оригінального паралельно-ієрархічного (ПІ) перетворення зображень [2] на основі CPU-орієнтованої апаратної платформи містить такі основні етапи:

1) завантажуємо зображення розмірністю n×m пікселів (де n – висота зображення, m – ширина зображення) в пам'ять системи;

2) елементи (пікселі), які мають нульове значення, відповідно до властивості ПІ перетворення не є інформативними. В процесі роботи алгоритму кількість нульових елементів значно перевищує кількість ненульових, тому, для прискорення роботи, числові дані про кожен ненульовий елемент будемо зберігати у власній структурі, що містить три поля (значення, рядок, стовпець);

3) сформуємо структуру розмірністю [n×m] (максимально можлива кількість ненульових елементів) і заповнимо її даними при кожен ненульовий елемент;

4) введемо дві змінні L та R (ліва та права границя), в межах яких ми працюємо з масивом даних. Ініціалізуємо L = 0, R – рівне кількості ненульових елементів;

5) основу ПІ-перетворення складають три обчислювальні операції: транспонування (T), G-перетворення (G), зсув (S). Детальніше етапи виконання кожної обчислювальної операції розглянуто нижче;

6) виконаємо операцію G-перетворення;

7) доки L < R – 1, тобто, доки для обробки не залишився один елемент, виконуємо послідовно три обчислювальні операції: T, G, S;

8) Елементи (0, L] – хвостові елементи, тобто результат роботи алгоритму.

Обчислювальна операція T:

A.1) в масиві даних [L, R) для кожного елемента змінюємо місцями значення рядка і стовпця;

A.2) змінюємо місцями значення n, m;

A.3) дані транспоновані, проте, щоб у масиві був правильний порядок, необхідно впорядкувати масив в межах [L, R).

Обчислювальна операція G-перетворення:

B.1) впорядковуємо масив в межах [L, R) використовуючи наступну логіку. При порівнянні: елемент А зустрічається раніше елемента Б, якщо значення рядка елемента А менше значення рядка елемента Б, або при їх рівності значення А менше значення Б.

B.2) для кожного рядка підраховуємо, яку кількість елементів він містить;

B.3) ініціалізуємо змінну sum = 0;

B.4) для кожного рядка, будемо переглядати зліва направо, і для кожного елемента обраховувати, який елемент він утворить у перетвореній матриці, що рівний добутку елемента на кількість ненульових елементів. Якщо він утворить ненульовий елемент, то записуємо його в рядок і змінюємо значення суми;

B.5) *R* = (кількість утворених ненульових елементів) + *L*;

B.6) *m* – рівне максимальній кількості ненульових елементів в нових рядках.

Обчислювальна операція S:

C.1) L = L + 1;

C.2) в масиві даних [L, R) для кожного елемента до значення стовпця додаємо значення рядка мінус 1 (індексація з 0). Таким чином, ми зсуваємо *i*-й рядок на *i* елементів вправо;

C.3) в масиві даних [L, R) знаходимо максимальне значення стовпця і присвоюємо *m* цьому значенню.

Реалізація на основі GPU-орієнтованої апаратної платформи

Алгоритм виконання оригінального ПІ перетворення зображень на основі GPU-орієнтованої апаратної платформи містить такі основні етапи:

1)–5) – ідентичні пунктам програмної реалізації на основі CPU-орієнтованої апаратної платформи;

6) скопіюємо дані на GPU;

7) виконаємо операцію G-перетворення на GPU (G_GPU);

8) доки L < R – 1 виконуємо послідовно три обчислювальні операції на GPU: T_GPU, G_GPU, S_GPU;

9) елементи (0, L] – хвостові елементи, результат роботи алгоритму, повертаємо цю частину масиву назад на CPU;

Обчислювальна операція T_GPU:

D.1) в масиві даних [L, R) для кожного елемента паралельно змінюємо місцями значення рядка і стовпця;

D.2) паралельно змінюємо місцями значення n, m;

D.3) логіка ідентична пункту A.3) програмної реалізації операції T на основі CPU-орієнтованої апаратної платформи, але сортування виконуємо паралельно, використовуючи метод бібліотеки thrust::sort() і власний компаратор.

Обчислювальна операція G-перетворення – G_GPU:

E.1) паралельно впорядковуємо масив в межах [L, R) використовуючи логіку при порівнянні, яка відповідає пункту B.1) програмної реалізації операції G-перетворення на основі CPU-орієнтованої апаратної платформи. Сортування виконуємо паралельно, використовуючи метод бібліотеки thrust::sort() і власний компаратор;

E.2) для кожного рядка паралельно підраховуємо, яку кількість елементів він містить;

E.3) ініціалізуємо змінну sum_gpu = 0;

E.4) для кожного рядка паралельно, будемо переглядати зліва направо, і для кожного елемента обраховувати, який елемент він утворить у перетвореній матриці, що рівний добутку елемента на кількість ненульових елементів, тобто

int tmp_val = (data[cur].val – sum_gpu) * (row_cnt[i] - j).

Якщо він утворить ненульовий елемент, то записуємо його в рядок і змінюємо значення суми:

sum_gpu += (data[cur].val – sum_gpu);

E.5) *R* = (кількість паралельно утворених ненульових елементів) + *L*;

E.6) *m* – рівне максимальній кількості ненульових елементів в нових рядках. Максимум в масиві знаходимо паралельно, з використанням thrust::reduce().

На основі описаних програмних реалізацій були отримані результати, які показані у таблиці 1. В якості вхідних даних використовувались відеоряди, які складаються з 2044 зображень профілю лазерного променя у кольоровій моделі Grayscale.

Таблиця 1 – Отримані результати

Розмірність зображення, пікселі	Час виконання на GPU, мсек	Час виконання на CPU, мсек
128 × 128	4500	400
256 × 256	5226	1076
512 × 512	10291	7450
1024 × 1024	15039	18900
2048 × 2048	23525	32809

Висновки

Таким чином, CPU-орієнтований алгоритм показує кращі результати при роботі із зображеннями малої розмірності. Проте, починаючи із зображень розмірністю 1024×1024 пікселі, GPU-орієнтований алгоритм починає перевершувати лінійний алгоритм. Це помітно із отриманих результатів, адже серія зображень 2048×2048 пікселя обробляється GPU-версією 23.525 секунд в порівнянні з 32.809 секундами CPU-версії, що доводить доцільність проведення описаних досліджень. Також, до переваг використання GPU-орієнтованої платформи можна віднести те, що дана сфера активно розвивається, створюючи адаптери з меншим часом передавання інформації як всередині GPU, так і на CPU, що безпосередньо впливає на швидкодію описаної програмної реалізації.

Список використаних джерел:

1. Яровой А.А., Яровой А.М. Теоретико-методологические и прикладные аспекты использования технологий визуализации для задач профилирования лазерных лучей [Электронный ресурс] : [Электронный журнал Национального исследовательского ядерного университета "МИФИ", Москва] / А.А. Яровой, А.М. Яровой // Научная визуализация. – 2010. – Том 2. – №3. – С. 50-72. – Режим доступу до журн.: http://sv-journal.com/2010-3/04/index.html.

2. A.A. Yarovyy, L.I. Timchenko, N.I. Kokriatskaia, S.V. Nakonechna, M.S. Mateichuk Organization of High-Performance Parallel-Hierarchical Computing Processes for Classification of Laser Beam Images – Development and application systems : Proceedings of the 12th International Conference on DAS-2014, May 15-17, 2014, Suceava, Romania – Suceava, Universitatea Stefan cel Mare Suceava, 2014. – p. 192-197.