C++ AMP (Accelerated Massive Parallelism) – бібліотека на основі DirectX 11 з відкритою специфікацією від компанії Microsoft для реалізації паралельних програм для гетерогенних систем. Для роботи необхідні операційна система Windows 7 або Windows Server 2008 R2 або вищі, з встановленим DirectX 11. Якщо у системі немає графічного адаптера, на якому може бути виконано код, то він буде запущений за допомогою центрального процесору і використовувати SSE інструкції. Для розробки потрібно середовище Microsoft Visual Studio 2012 або вище. Бібліотека є простою у використанні та апаратно незалежною, на відміну від, наприклад, CUDA. Найбільш «вузьке» з точки зору продуктивності місце – процес копіювання даних с центрального процесора на графічний та навпаки. Тому при переносі алгоритмів важливо враховувати не тільки можливість незалежного виконання потоків, але й кількість даних, що треба скопіювати. Крім того, бібліотека має багато обмежень: у типах та структурах даних, покажчиках, операціях тощо[1], що викликає деякі труднощі при реалізації алгоритмів:

* Для зберігання вхідного зображення ­ достатньо матриці типу uchar (1 байт) розміром height\*width\*С, де С - кількість каналів. Щоб уникнути зайвого копіювання даних, канал зберігався в матриці типу unsigned int розміром int size = (rows\*cols+3)/4, а доступ реалізовано через безпосередній доступ до бітів.
* структура даних, що описує дерево, в оригінальному алгоритмі, зберігається у вигляді матриці, кожний рядок якої містить вузол. В залежності від об’єму навчальної вибірки та максимальної глибини дерева, матриця є розрідженою. У [2] дерева зберігаються у структурі даних для текстур також у розрідженому вигляді. Щоб уникнути цього, в окремій матриці зберігаються непорожні вузли, в окремій – «покажчики» на них. C++ AMP також містить тип texture, однак її максимальний розмір є більшим за розмір множини вузлів.
* Для зберігання листя створені матриці з даними та покажчиками (рис.1): такий підхід пов’язаний з неможливістю використання складних структур даних, масивів з масивів змінного розміру, але тільки прямокутних масивів. Щоб отримати дані листа у вигляді цілих чисел, ймовірності pi збільшувалися у 1000 разів, співвідношення сторін ri – у 10.
* Вихідні дані представлені у вигляді 2 матриць типу int розміром height\*width\*c, де с – кількість класів. В кожному каналі містяться результати голосування для окремого класу. Перша матриця – отриманий Хаф простір, друга – співвідношення сторін прямокутника, що обмежує об’єкт. Задля зменшення обсягу копійованих даних, перші два біта містять акумульовану суму співвідношень, другі два – кількість голосів.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Індекс | 0 |  |  | і1 | | і2 | |  |  |  |  | і3 |  | і4 |  |  | |  |  | |  |  |
| Дерево 0 | Лист 0 | | | | | | Лист 1 | | | | | | | Лист 2 | | | … | | | | | |
| … | … | | | | | | | | | | | | | | | | | | | | | |
| Дерево N | Лист 0 | | | | Лист 1 | | | | | | | Лист 2 | | | | | | | | … | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Дерево 0 | 0 | і2 | і4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| … | … | | | | | | | | | | | | | | | | | | | | | | | | |
| Дерево N | 0 | і1 | і3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Рис. 1 Організація листя у вигляді матриць

Організація пам’яті на графічних процесорах відрізняється у різних виробників, але загальна схема[3] зображена на рис. 2. Відповідно до неї, на GPU швидко виконується велика кількість незалежних один від одного математичних обчислень, проте, на відміну від CPU, більш повільно виконуються умовні оператори, цикли.

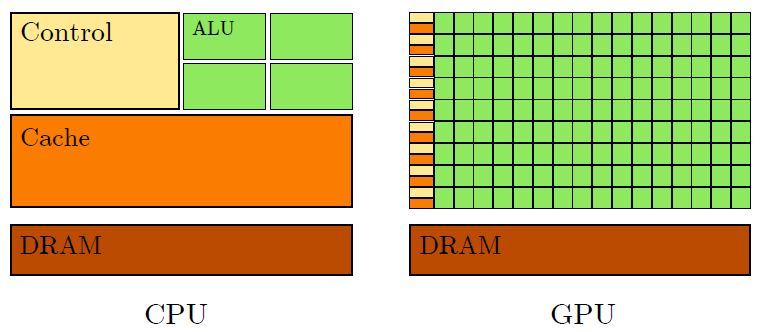


Рис. 2 Схема пам’яті CPU та GPU

У перенесеному на GPU алгоритмі голосування відбувається паралельно для кожного пікселя в кожнім дереві, тобто у (tree\_count\* height\*width) потоках.

1. **Lingli Zhang**. *restrict(amp) restrictions part 0 of N – introduction* – Режим доступу: <http://blogs.msdn.com/b/nativeconcurrency/archive/2011/12/19/restrict-amp-restrictions-part-0-of-n-introduction.aspx>
2. **Toby Sharp**. “Implementing decision trees and forests on a GPU”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 595–608
3. **А.В. Григорьев, И.С. Еремеев, М.И. Алексеева** Учебное пособие  
   Параллельное программирование с использованием технологии CUDA – Електронний ресурс – Режим доступу: <http://edu.chpc.ru/cuda/mainse2.html#x5-40001.2>
4. **D. H. Ballard**. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*