Реконструкція 3D-моделей реальних об’єктів методом RBF з використанням GPU

Бабков В.С.
Донецький національний технічний університет
victor@babkov.name

Abstract

Babkov V. Reconstruction of real object 3D model by RBF method on GPU. *Count and size cost of 3D reconstruction method by RBF are estimated. Possibility hardware acceleration of reconstruction process by GPU are discussed. Conclusion about requirement for modified algorithms are formulated.*

Вступ

У наш час у багатьох галузях науки і техніки важливу роль відіграє використання тривимірних комп’ютерних моделей, побудованих за результатами дослідження реальних об’єктів складної форми. Подібна задача виникає, наприклад, у таких випадках:

- пошук дефектів у структурі об’єктів;
- вивчення внутрішньої структури об’єкту без його руйнації;
- відновлення об’єкту за неповними даними;
- побудова тривимірних моделей рухливих об’єктів у реальному часі;
- оцінка, реконструкція та проектування великих промислових об’єктів та ділянок місцевості;
- обробка результатів наукових експериментів та складних обчислень.

Існує велика кількість методів отримання проекційних даних для побудови тривимірних моделей [1-3]. Загальна риса цих методів – видача результатів сканування у вигляді „хмари” точок, що описують поверхню об’єкту або системи взаємопов’язаних об’єктів. Задача реконструкції 3D моделей реальних об’єктів за такими даними найчастіше зводиться до задачі інтерполяції у тривимірному просторі [4]. Для розв’язання такої задачі широко застосовується метод RBF (radial basis function) [5].

У роботі [6] було показано, що цей метод і його модифікації:

- метод RBF з мінімізацією кількості центрів інтерполяції
- метод компактної RBF;
- метод швидкої RBF;
- метод декомпозиції площин;

мають значну обчислювальну та просторову складність, тому актуальним є прискорення процесу реконструкції, як за допомогою алгоритмічних, так і за допомогою архітектурних засобів. Питання апаратного прискорення процесу реконструкції розглядається у багатьох сучасних публікаціях [7-11].

У статті ставиться задача проаналізувати можливі варіанти реалізації реконструкції за допомогою методу RBF з використанням спеціалізованих архітектур та архітектур загального призначення і сформулювати перелік вимог до алгоритмів реконструкції щодо їх ефективної апаратної реалізації.

Аналіз архітектурних засобів розв’язання задачі реконструкції

За результатами експериментальних досліджень проведених автором, та узагальнення результатів, отриманих в інших публікаціях [6, 15-17] побудовані залежності часу реконструкції та обсягу пам’яті, від складності об’єктів (див. рис. 1-2). Залежності побудовані у логарифмічному масштабі через значний діапазон зміни залежної та незалежної змінної.

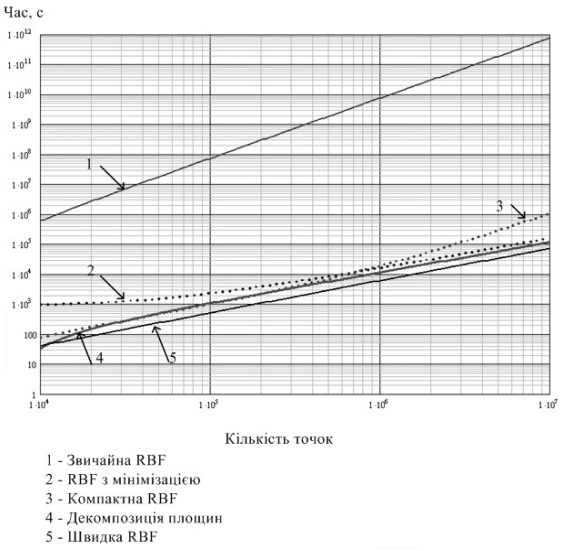


Рисунок 1 – Залежність часу обробки від кількості точок об’єкту

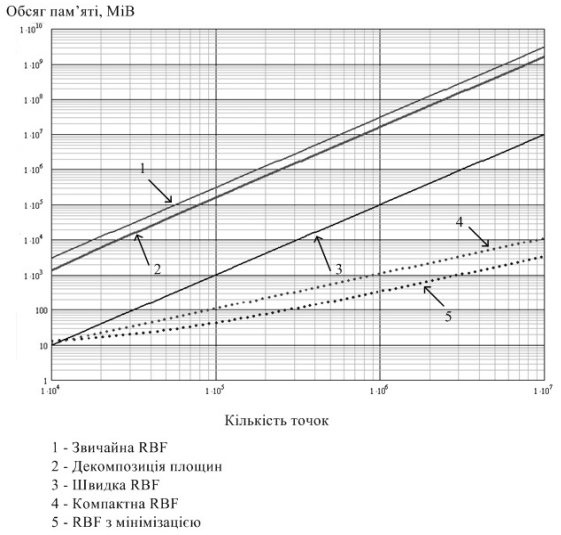


Рисунок 2 – Залежність обсягу пам'яті від кількості точок об'єкту

За аналізом алгоритмів реконструкції, що був виконаний у [6], побудована порівняльна таблиця обчислювальної та просторової складності алгоритмів (див. табл. 1).

У таблиці 1 використані наступні позначення:

RBF – прямий метод;

RBFm – метод з мінімізацією кількості центрів інтерполяції;

CRBF – метод „компактних" RBF;

FRBF – метод швидкого обчислення;

SLICE – метод декомпозиції площин;

Етап 1 – етап обчислення RBF;

Етап 2 – етап розв'язання системи;

Етап 3 – етап розрахунку поверхні;

Етап 4 – обсяг матриць у пам'яті.

Таблиця 1. Обчислювальна та просторова складність алгоритмів

Етап	Алгоритм				
	RBF	RBFm	CRBF	FRBF	SLICE
1	$O(n^2)$	$O(n^2)$	$O(n\log n)$	$O(n^2)$	$O(s^2)$
2	$O(n^2)$	$O(n^2)$	$O(n^{1.2...1.5})$	$O(n\log n)$	$O(s^2)$
3	$O(mn)$	$O(mn)$	$O(m\log n)$	$O(m+n\log n)$	$O(mn)$
4	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n^2)$	$O(s^2)$

n - кількість проекційних точок;

m - кількість точок у розрахованій поверхні;

s - середня кількість точок в одній площині ($s << n$).

Аналізуючи розглянуті методи реконструкції, можна зробити наступні висновки. Не дивлячись на те, що існує багато методів реконструкції, які зменшують обчислювальну та просторову складність алгоритмів, кожний з них має свої недоліки [6]: вузька сфера застосування, чутливість до типу вхідних даних, завеликий обсяг пам'яті, що використовується та ін.

Крім того, хоча відносне зменшення обчислювальної та просторової складності, за даними таблиці 1, досить значне, експериментальні дослідження показують, що при реалізації методів на звичайних ПЕОМ, часові характеристики реконструкції (100-300 с для кількості точок 20000-100000) не дозволяють виконувати реконструкцію у реальному часі. Наприклад, для реконструкції складних систем об'єктів, або об'єктів, що динамічно змінюються у часі.

Як було описано вище, процес реконструкції поверхні об'єкта за допомогою одного з методів із використанням RBF можна представити у вигляді чотирьох послідовних етапів: див. рис. 3.

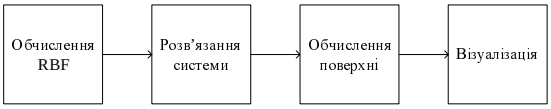


Рисунок 3 – Послідовність етапів реконструкції

У літературі не вдалося знайти інформації про наявність єдиних архітектурних рішень для розв'язання задачі реконструкції в цілому. Однак існують архітектурні засоби, що реалізують різні етапи даного процесу як для здійснення безпосередньо реконструкції, так і для використання у інших системах, що використовують методи RBF [7].

Оскільки методи, що використовують RBF, застосовуються у багатьох прикладних задачах, то давно відомі архітектурні реалізації для їх обчислення у вигляді систолічних масивів [8] (див. рис. 4). Такі масиви застосовувалися у складі багатьох спеціалізованих НВІС, що призначені, в тому числі, і для обробки зображень, у якості RBF-препроцесора [15].

У літературі описані різні комбіновані методи реалізації етапів реконструкції [9] з використанням CPU та GPU загального призначення.

Типова архітектура GPU, що використовуються для реалізації процесу реконструкції, наведена на рис. 5. Такій архітектурі відповідають сучасні відеопроцесори S3, TSMC та ін., що використовуються у складі графічних плат NVidia 6600/6800, 8800, ATI Radeon X700/1300 та ін. [16, 17].

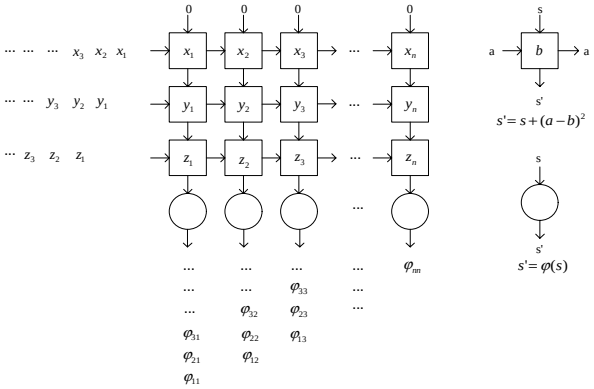


Рисунок 4 – RBF-препроцесор на основі систолічного масиву

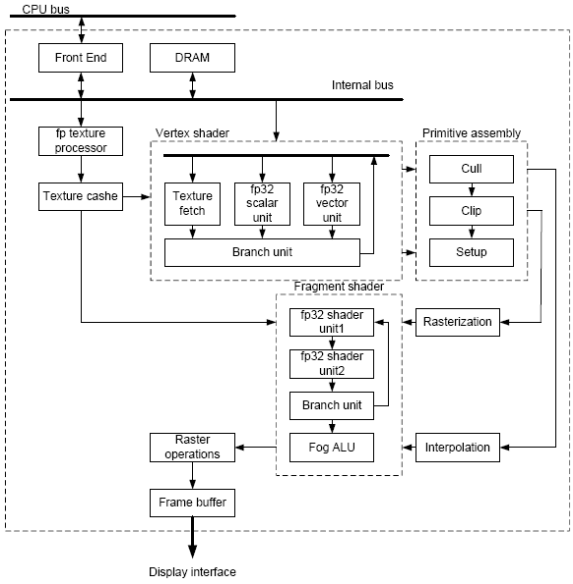


Рисунок 5 – Архітектура типового GPU

Типовий GPU складається з наступних блоків.

CPU bus – шина процесора, через яку здійснюється обмін даними між CPU та GPU.

Front End - блок, що приймає від процесора команди та дані і розташовує їх у внутрішній пам'яті;

DRAM – блок пам'яті, який використовується для зберігання даних про об'єкт (координати, кольорові характеристики, текстури, програми роботи вбудованих процесорів - шейдерів),

Texture processor – блок, що забезпечує попередню обробку текстур.

Texture cashe – кеш, який розділяється між процесором вершин та фрагментів та зберігає блоки текстур;

Vertex shader – процесор вершин, що забезпечує перетворення координат об'єкту у дисплейні координати.

Primitive assembly – блок обробки примітивів, що забезпечує перетворення вершин у точки, лінії та полігони, виконує з'єднання елементів та підготовку до растерізації.

Rasterization – блок растерізації, що для кожного елементу визначає видиму частину.

Interpolation – блок інтерполяції.

Fragment shader – процесор фрагментів, що забезпечує обробку блоків пікселів (накладання текстур).

Raster operation – блок растрових операцій.

Frame buffer – буфер для зберігання поточного кадру сцени.

Процесор фрагментів та процесор вершин у складі GPU являють собою паралельні блоки обробки даних з архітектурою SIMD. Наприклад, у процесорі карти ATI Radeon X550, що виконаний на основі ASIC RV370XT, кількість процесорних елементів складає 128 [16]. Можливість паралельної обробки даних у форматі з плаваючою комою може бути використана не тільки для виконання специфічних графічних операцій, а й для виконання довільних операцій, пов'язаних із обчисленням функцій, знаходженням матрично-векторних добутків та ін. [9].

Дослідження варіантів реалізації алгоритмів реконструкції на CPU та GPU

Загальна схема реалізації етапів реконструкції на основі одночасного використання CPU та GPU наведена на рис. 6.

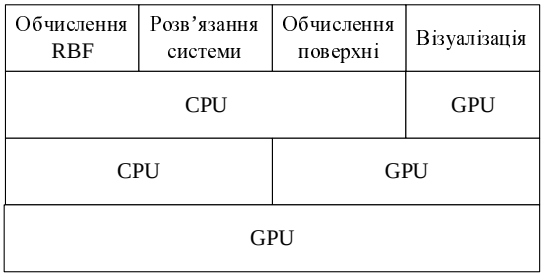


Рисунок 6 – Варіанти апаратної реалізації процесу реконструкції

Реалізація CPU(1,2,3)+GPU(4) (у дужках вказано номери етапів, на яких працює відповідний пристрій) найчастіше використовується при реалізації методів реконструкції у тих випадках, коли немає потреби в реконструкції у реальному часі. Експериментальні оцінки на рис. 1-2 були отримані саме при використанні повністю програмної реалізації алгоритмів на CPU загального призначення. При цьому етап

візуалізації реалізовувався за допомогою стандартних засобів GPU з використанням бібліотеки Open GL. У літературі описано багато методів прискорення етапу візуалізації поверхневих та об'ємних моделей при використанні алгоритмів ray-tracing, 3D texture mapping, shading та ін. на типових GPU [10].

Реалізації CPU(1,2)+GPU(3,4) та GPU(1,2,3,4) описані у [10]. У першому варіанті на CPU покладаються задачі обчислення RBF та розв'язання системи, а на GPU - задачі розрахунку поверхні та візуалізації. Схема процесу обробки для даного випадку наведена на рис. 7.

У другому випадку fragment-процесор використовується спочатку для обчислення функції та розв'язання системи, а потім – для пошуку поверхні та її візуалізації. Схема процесу обробки для даного випадку наведена на рис. 8.

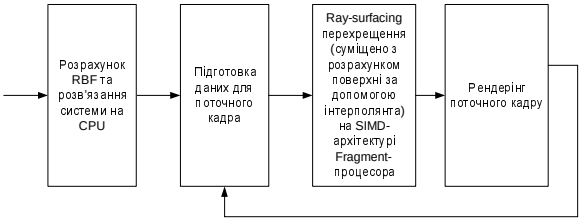


Рисунок 7 – Схема процесу обробки для варіанту CPU(1,2)+GPU(3,4)

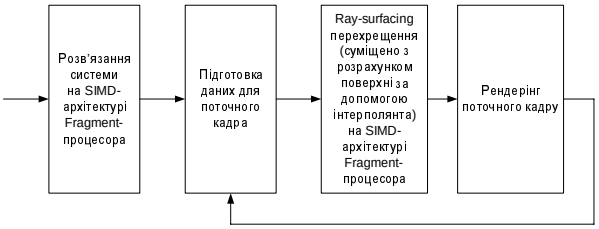


Рисунок 8 – Схема процесу обробки для варіанту GPU(1,2,3,4)

Аналіз експериментальних даних при реалізації реконструкції за схемами на рис. 7-8 (див. рис. 9) дозволяє зробити висновок, що швидкість обробки даних є найбільшою при використанні CPU на етапах обчислення функції та розв'язання системи і GPU на етапах побудови поверхні та візуалізації. Відмова від попередньої обробки на CPU зменшує швидкість обробки, оскільки GPU має такі недоліки:

- неефективна організація пам'яті для зберігання списочних структур (дерев), що використовуються у багатьох методах RBF;
- неможливість реалізації рекурсивних алгоритмів на GPU (в усякому випадку на тих, що існують зараз) [16].

На рис. 9 використані такі позначення:
T1 – варіант CPU(1,2,3)+GPU(4);
T2 – варіант CPU(1,2)+GPU(3,4);
T3 – варіант GPU(1,2,3,4).

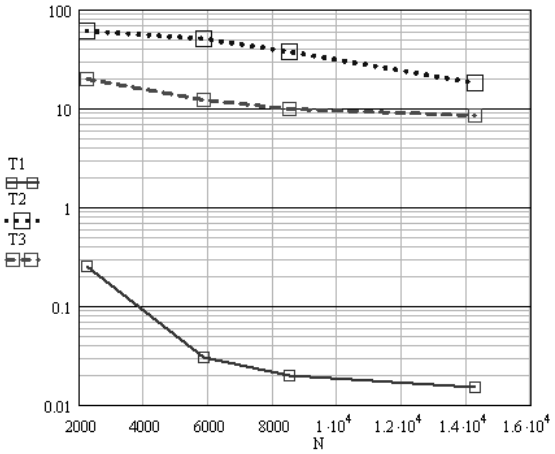


Рисунок 9 – Залежність швидкості реконструкції та візуалізації (fps) від кількості точок об'єкту для різних варіантів апаратної реалізації

На рис. 10 показані приклади результатів реконструкції тривимірних моделей за проекційними даними за допомогою програмної системи [18].

У роботі [11] описана спроба повної реалізації методу FMM на GPU і за результатами дослідження зроблені висновки, які підтверджують недоліки GPU при реалізації методів, що використовують списочні структури даних та рекурсивні алгоритми.

При цьому використання GPU тільки на етапі обчислення поверхні та знаходження матрично-векторних добутків дає приріст швидкості у 5..60 разів у порівнянні з використанням лише CPU [11].

Висновки

Таким чином, проаналізувавши, архітектурні засоби, спрямовані на прискорення процесу реконструкції, можна зробити наступні висновки.

Використання сучасних GPU дозволяє ефективно прискорювати не всі етапи реконструкції. Пов'язано це з тим, що алгоритми та структури даних, що використовуються у різних методах реконструкції та архітектури сучасних GPU не адаптовані один до одного.

Зокрема, відомі алгоритми реконструкції мають такі недоліки (відносно їх реалізації на GPU):

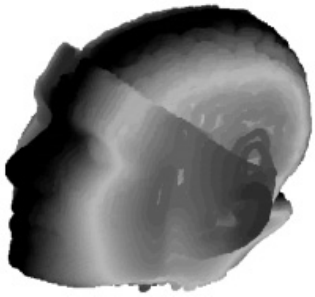
- використання складних динамічних структур даних (дерев, списків);

- великий обсяг матриць (10^2..10^4 МіВ);

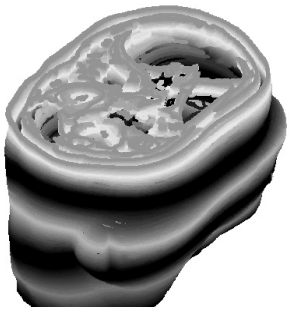
- використання рекурсії;

- послідовний характер методів (алгоритми погано адаптовані до реалізації на SIMD-архітектурах)

Таким чином, мета подальшого дослідження – це модифікувати відомі методи реконструкції у напрямку подальшого зменшення обчислювальної та просторової складності з можливістю ефективної реалізації з використанням апаратних засобів прискорення.



а)



б)

в)

Рисунок 10 – Приклади результатів реконструкції: фантом голови людини (а), позвонок (б), Standford Bunny (в)

Література

1. Клюев В.В. Неразрушающий контроль и диагностика. - М.: Машиностроение, 1995 – 370 с.
2. Хермен Г. Восстановление изображений по проекциям: Основы реконструктивной томографии. Пер. с англ. – М.: Мир, 1983 – 352 с.
3. Мельников С.Р. Лазерное сканирование: новый метод создания трехмерных моделей местности и инженерных объектов // Информационный бюллетень „ГИС-Ассоциации” № 2 – 3, 2001
4. P. Alfeld, Scattered data interpolation in three or more variables, in Mathematical Methods in Computer Aided Geometric Design, T. Lyche and L. Schumaker, eds, Academic Press, 1989, pp. 1-33
5. J. C. Carr et al Reconstruction and Representation of 3D Objects with Radial Basis Functions ACM SIGGRAPH, 12-17 August 2001, pp67-76,
6. Бабков В.С., Ивашковец Е.В. Проектирование многофункциональной программной системы для реконструкции трехмерных объектов в медицинской практике Сборник трудов третьей международной научно технической конференции молодых ученых и студентов «Информатика и компьютерные технологии» 11-13 декабря 2007 г. Донецк, ДонНТУ, с. 285-287
7. El Shishiny et al., US 7.272.264 B2 Sep. 18, 2007 System and method for hole filling in 3D models
8. Broomhead D.S. et al A systolic array for nonlinear adaptive filtering and pattern recognition. IEEE International Symposium on Circuits and Systems, 1-3 May 1990 Los Angeles, US, vol. 2 pp 962-965
9. Fryazinov O., Pasko A GPU-based real time FRep ray casting GraphiCon, Moscow, June 23-27, 2007
10. Westermann, R. and T. Ertl. “Efficiently Using Graphics Hardware in Volume Rendering Applications”, SIGGRAPH, July 1998, pp.169–177.
11. Gumerov N.A. Fast summation of potential the FMM on the GPU, AstroGPU 2007,
12. Morse, B.S. et al Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, SMI 2001 International Conference – May 2001, P: 89-98
13. Beatson R.K., Newsam G. N., Fast evaluation of radial basis function: I, Computers Mathematics and Applications Vol. 24, No. 12, pp. 7-19, 1992
14. Qiang W. et al Surface rendering for parallel slice of contours from medical imaging. Computing in science & engineering, Vol. 9, No. 1, 2007, pp 32-37
15. БИС для распознавания образов и обработки изображений. Пер. с англ./Под ред. К. Фу. - М.: Мир, 1988. – 248 с.
16. GPU Gems 2, Chapter 30. The GeForce 6 Series GPU Architecture, NVidia Corp., 2005
17. NVIDIA GPU Programming Guide 2.5.0, 2006
18. В.С.Бабков Реконструкція 3D-моделей органів в комп’ютерній томографії при обмеженому об’ємі вхідних даних. Наукові праці ДонНТУ. Серія “Проблеми моделювання та автоматизації проектування динамічних систем”. Випуск 52 – Донецьк: ДонНТУ. 2002. - 100-105 сс.