Инженерный вестник Дона, №4 (2015)
ivdon.ru/ru/magazine/archive/n4y2015/3322

Применение кривой Гильберта для обхода точек расчётной области в задачах обработки изображений

Р.В. Арзуманян

Институт компьютерных технологий и информационной безопасности Южного федерального университета, Таганрог

Аннотация: Работа посвящена исследованию применимости кривой Гильберта для обхода точек расчётной области в задачах цифровой обработки изображений на графических процессорах с поддержкой вычислений общего назначения. Исследование приводится на примере эксперимента по применению Гауссова размытия, реализованного для построчного и изменённого порядков обхода. Фильтр реализован как ядро стандарта гетерогенных вычислений OpenCL и применяется для обработки 8-битных изображений в оттенках серого. Эксперимент проводится для графических карт с дискретной памятью и с использованием памяти, разделяемой графическим и центральным процессорами. Анализируется влияние порядка обхода точек изображения на обращения к глобальной памяти графического процессора и влияние на работу кэша.
Ключевые слова: Гильбертова кривая, шаблон доступа к памяти, графический процессор, OpenCL.

Введение. В последнее десятилетие наметились изменения в темпах прогресса в вычислительной технике – переход от экстенсивного роста частот к интенсивному наращиванию мощности [1] путём горизонтального масштабирования вычислительных систем. В этой связи, всё большее значение приобретает разработка локальных алгоритмов – тех, которые бы использовали для передачи данных низкоуровневые быстрые локальные каналы передачи данных [2]. Это необходимо потому, что скорость обмена информацией значительно ниже, чем скорость её обработки на современных компьютерах [3]. Ускорители вычислений с архитектурой массового параллелизма особенно нуждаются в таких алгоритмах, поскольку массовые обращения к глобальной памяти на таких устройствах приводят к значительным просадкам производительности [4].

В некоторых работах [5] рассматривается использование двумерных кривых, имеющих свойство заполнения пространства. Такие кривые носят название кривых Пеано. Одним из частных случаев кривых Пеано является

Гильбертова кривая – фрактальная кривая, заполняющая пространство. Алгоритм построения кривой представлен на рис. 1.

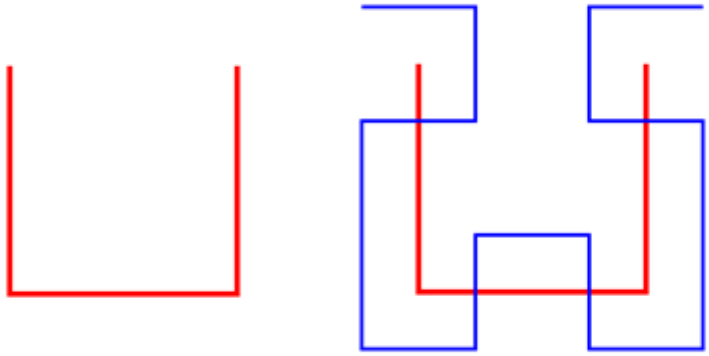

Рис. 1. - Алгоритм построения кривой Гильберта.

Все кривые 1го порядка в составе кривых более высокого порядка рекурсивно заменяются на кривые 2го порядка. Данная кривая обладает свойством локальности – звенья кривой с близкими номерами имеют близкие по значению пространственные координаты [6]. Если расчётная область представляет собой квадрат, то её можно заполнить Гильбертовой кривой произвольного порядка [7].

Свойство локальности хорошо подходит для реализации алгоритмов на графических ускорителях с поддержкой вычислений общего назначения (далее GPGPU) – обращения к памяти из соседних потоков будут происходить к соседним адресам. Так как объём кэшей на GPGPU и ускорителях вычислений небольшой, а обращения к памяти - медленные, то использование алгоритмов с хорошим шаблоном доступа к памяти может существенно увеличить производительность.

Однако у предлагаемого способа есть недостаток – вычисление координаты точки по длине кривой в заданной точке является для

графического процессора (далее GPU) неподходящей задачей, так как требует последовательных вычислений в цикле [8].

В данной работе рассмотрено влияние порядка обхода точек расчётной области на производительность алгоритмов обработки изображений. В качестве порядков обхода рассматриваются обход растром и в порядке, определяемом узлами Гильбертовой кривой. В качестве алгоритма обработки изображений рассматривается применение Гауссова размытия.

Постановка задачи. В качестве задачи в данной работе рассматривалась фильтрация чёрно-белых изображений с глубиной цвета 8 бит при помощи Гауссова размытия. Данный фильтр хорош тем, что сочетает в себе как удачные (по строкам), так и неудачные (по столбцам) способы обращения к памяти [9], а также то, что его можно реализовать в виде свёртки, что даст примерную оценку применимости предполагаемого алгоритма для свёрточных методов обработки изображений.

В качестве тестовых систем рассматривались компьютеры, оснащённые видеокартой Radeon 7660D:

- Архитектура Very Long Instruction Word 4 (далее VLIW4).
- Не имеет собственной памяти, использует ОЗУ совместно с центральным процессором (далее CPU).
- Шина доступа – 2 канала по 64 бита.
- Эффективная частота памяти – 1600 МГц.

Radeon 8670M:

- Архитектура Graphics Core Next (далее GCN).
- 1 Гбайт выделенной памяти.
- Шина доступа – 128 бит.
- Эффективная частота памяти – 1800 МГц.

Цель данной работы – ответить на вопросы:

- Может ли использование кривой Гильберта для обхода точек расчётной области привести к ускорению работы алгоритмов свёрточной обработки изображений?
- Если да, то при каких условиях возможно получить ускорение?

Описание экспериментов. В данной работе был произведён ряд экспериментов по фильтрации изображений при помощи Гауссова размытия. Эксперименты производились над изображениями размером 512x512, 1024x1024, 2048x2048 и 4096x4096 пикселов фильтром с радиусом 5 пикселов. Фильтр Гаусса реализован в «двухпроходной» версии, когда фильтрация проводится сначала по вертикали, а затем по горизонтали. Алгоритм реализован как ядро [10, 11] OpenCL. Объекты памяти выделялись в двух режимах – разделения с CPU (CL_MEM_ALLOC_HOST_PTR) и в варианте по умолчанию, когда объект памяти выделяется из объёма памяти устройства (CL_MEM_READ_WRITE).

Далее на графиках, эксперименты под названием «Raster order» проводились следующим образом:

- Координаты точки расчётной области читаются из объекта памяти, используется растровый порядок обхода.
- Один поток обрабатывает один пиксел изображения.

Эксперименты под названием «Hilbert order» проводились следующим образом:

- Координаты точки расчётной области читаются из объекта памяти, используется порядок обхода, задаваемый Гильбертовой кривой.
- Один поток обрабатывает один пиксел изображения.

Эксперименты под названием «Fast raster» проводились так:

- Координаты точки расчётной области поток получает от планировщика в виде индекса.
- При вертикальном проходе один поток фильтрует строку из 4 пикселов (видеокарта Radeon 8670) или 16 пикселов (видеокарта Radeon 7670).
- При горизонтальном проходе один поток фильтрует один пиксел.

Эксперименты под названием «Fast Hilbert» проводились так:

- Поток получает индекс от планировщика. Данный индекс в заданной точке равен длине кривой Гильберта. На основе длины вычисляется положение расчётной точки.
- Один поток обрабатывает один пиксел.

Анализ результатов экспериментов.

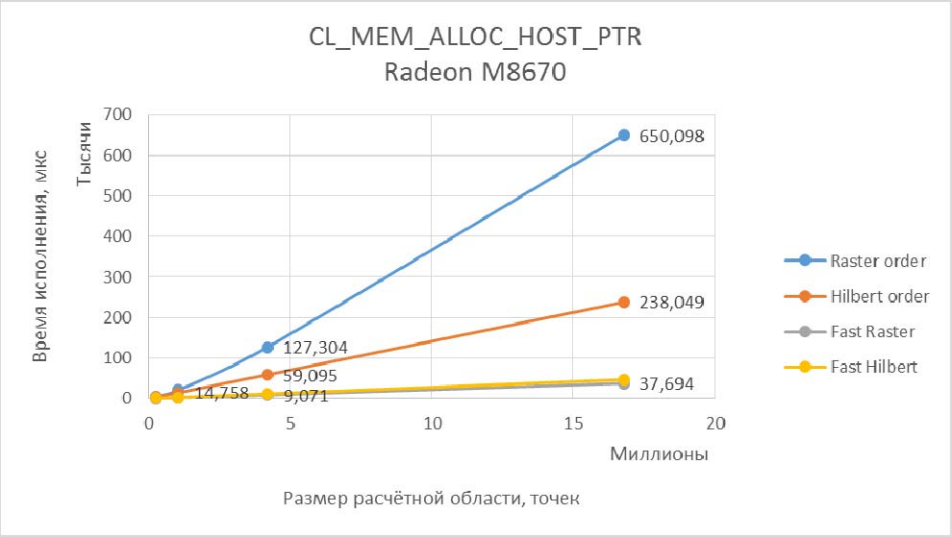

Рис. 2. - Результаты карты Radeon 8670M в режиме с разделяемой памятью

Режим разделения памяти между CPU и GPU для карты Radeon 8670M (рис. 2) осуществляется через дупликацию данных по шине PCI-Express, поэтому обращения к памяти в данном случае являются более

дорогостоящими, чем обычно. Применение кривой Гильберта в данном случае даёт очевидный результат – скалярные ядра фильтрации работают с той же скоростью, что и более быстрые векторные даже с учётом того, что ядро тратит много инструкций на определение двумерных координат точки расчётной области по значению длины кривой в точке.

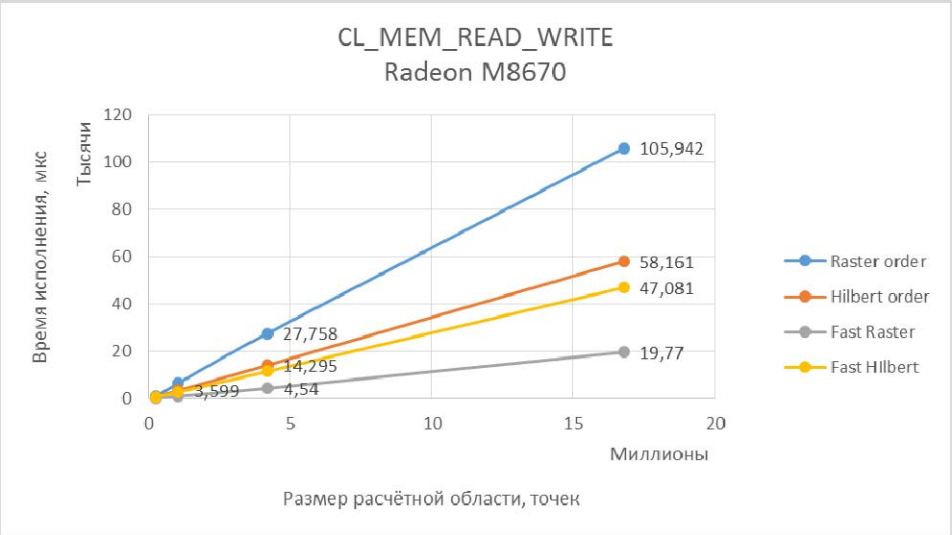

Рис. 3. - Результаты карты Radeon 8670M в режиме с использованием памяти видеокарты

Данный сценарий (см. рис. 3) наиболее благоприятен с точки зрения обращений к глобальной памяти GPU, поскольку используется относительно быстрая набортная память. В данном случае хорошо виден основной недостаток применения кривой Гильберта для обхода точек расчётной области – большие затраты на вычисление двумерных координат точки по длине кривой. Векторизованное ядро, использующее растровый порядок обхода, выполняет фильтрацию в 2.5 раза быстрее за счёт того, что получает координаты расчётной точки в виде индекса потока от планировщика.

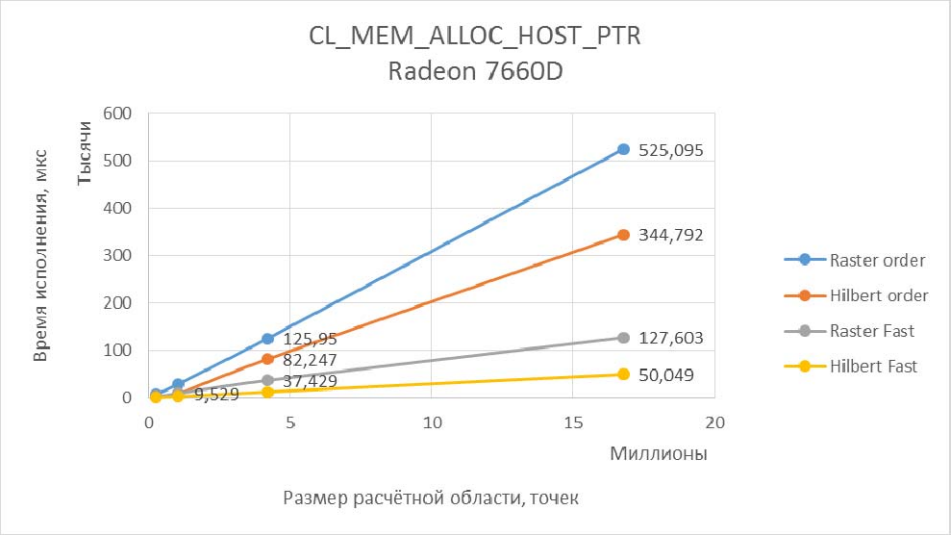

Рис. 4. - Результаты карты Radeon 7660D в режиме с разделяемой памятью

Видеокарта Radeon 7660D собственной памяти не имеет, и использует в качестве таковой оперативную память CPU. Поэтому интересным вариантом выделения памяти является фактическое разделение участка физической памяти из объёма ОЗУ при помощи передачи прав владения (отражения одного физического адреса в адресные пространства CPU и GPU). Результаты эксперимента, представленные на рис. 4 показывают, что обращения к памяти в таком случае, как правило, более медленные, поскольку дополнительная нагрузка ложится на контроллер памяти (обслуживает обращения от CPU и GPU) и драйвер видеокарты (отображает адресное пространство). Подобная ситуация является «благоприятной» для использования кривой Гильберта для обхода точек расчётной области, что подтверждается результатами эксперимента. Такой сценарий разделения физической памяти интересен так же тем, что позволяет избежать операций чтения-записи и дублирования данных. Он часто используется в мобильных устройствах, где GPU практически всегда использует память совместно с CPU.

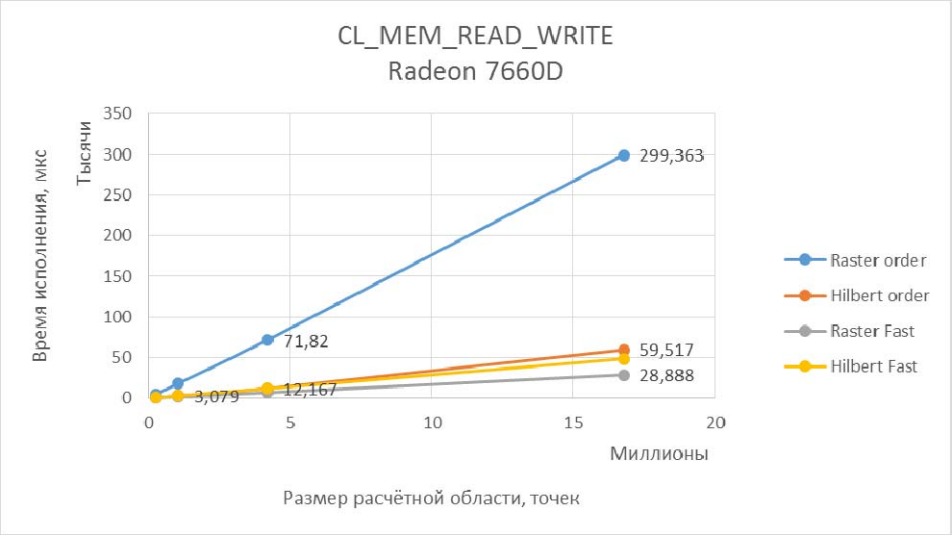

Рис. 5. - Результаты карты Radeon 7660D в режиме с использованием памяти видеокарты

Эксперимент, результаты которого представлены на рис. 5 в целом похож на эксперимент с теми же условиями, но проведённый на карте Radeon 8670M за тем исключением, что обращения к памяти более медленные, поэтому разница во времени между быстрым ядром, использующим растровый порядок обхода, и ядром, использующим кривую Гильберта, не столь велика.

Анализ результатов работы ядер проводился в профилировщике AMD CodeXL. Разработчиком данного профилировщика является компания AMD (USA, California), которая так же является производителем видеокарт, на которой проводились эксперименты. Благодаря этому из профилировщика можно получить доступ к низкоуровневой информации о производительности ядер OpenCL, собранной с применением аппаратных средств сбора информации, которые вносят минимальные изменения в результаты производительности.

Анализируя время работы ядер можно прийти к выводу, что применение кривой Гильберта в качестве порядка обхода расчётной области позволяет получить выигрыш в производительности, т. к. благодаря свойству локальности кривой, можно организовать хороший шаблон доступа к памяти, минимизирующий количество обращений к глобальной памяти устройства. В пользу данного предположения говорит значение счётчика производительности WriteSize (Объём записанных данных), получаемое при помощи профилировщика

Таблица 1.

Результаты профилировки ядер OpenCL

Название эксперимента	Объём записанных данных, Кбайт	Попаданий в кэш, %
GaussFilterY – raster order	11090	85.70
GaussFilterX – raster order	12073	86.50
GaussFilterY – Hilbert order	1032	54.91
GaussFilterX – Hilbert order	1024	56.06

Анализируя данные из таблицы 1, ясно что при использовании растрового порядка обхода точек расчётной области, происходит значительно больше обращений к глобальной памяти устройства, что приводит. Так же стоит обратить внимание на то, что использование растрового порядка обхода позволяет контроллеру памяти значительно лучше предсказывать обращения [12-13], и увеличивает количество предсказаний (и, соответственно, попаданий в кэш).

Таким образом, использование кривой Гильберта не является неким универсальным средством повышения производительности. В поставленном эксперименте выигрыш достигается только за счёт меньшего количества

обращений к глобальной памяти на запись, в то время как чтение происходит (судя по меньшему кэш-хиту), медленнее.

Более того, время расчёта положения точки на основе длины кривой Гильберта занимает значительное время. Если это время сопоставимо со временем исполнения основного кода ядра, то о выигрыше в производительности речи идти не может.

Заключение. В данной работе был рассмотрен вариант применения Гильбертовой кривой для обхода точек расчётной области применительно к задачам свёрточной обработки изображений. Данный подход продемонстрировал хороший результат с точки зрения увеличения производительности благодаря оптимизации обращений к глобальной памяти устройства.

Необходимо отметить, что применённый подход не является универсальным с точки зрения оптимизации производительности. Так, уменьшается процент попаданий в кэш из-за того, что использование кривой Гильберта для обхода точек расчётной области хуже обрабатывается системой кэшей.

По результатам работы можно сделать вывод о том, что применения кривой Гильберта для обхода точек расчётной области оправдано в случае, если реализуемый алгоритм интенсивно обращается к глобальной памяти и время обращения к ней достаточно велико (эксперименты с разделяемой памятью в режиме CL_MEM_ALLOC_HOST_PTR). В таком случае сложность расчёта координаты точки по длине кривой в точке компенсируется более оптимальной работой с глобальной памятью.

Литература

1. Гулякович Г. Н., Северцев В.Н., Шурчков И.О. Перспективы и проблемы полупроводниковой наноэлектроники // Инженерный вестник Дона. 2012. №2. URL: ivdon.ru/ru/magazine/archive/n2y2012/790

2. Воеводин В.В., Воеводин Вл. В. Спасительная локальность суперкомпьютеров // Открытые системы. - 2013. - №9. - С. 12-15.

3. Пономарева Е.И. Совершенствование процесса обработки данных при помощи облачных вычислений // Инженерный вестник Дона. 2012. №1. URL: ivdon.ru/ru/magazine/archive/n1y2012/628

4. Гервич Л.Р., Штейнберг Б.Я. Программирование экзафлопсных систем // Открытые системы. - 2013. - №8. – С. 26-29.

5. Уоррен Г.С.-мл. Алгоритмические трюки для программистов. - 2-е изд. – М.: Вильямс, 2013. - 512 с.

6. Александров П.С. Введение в общую теорию множеств и функций. – М.: Огиз-Гостехиздат, 1948. - 413 с.

7. Кривые Гильберта и Серпинского, или снова рекурсия // 1september.ru URL: inf.1september.ru/1999/art/zlat1.htm (дата обращения: 28.10.2015).

8. Кушниренко А.Г., Лебедев Г.В., Скворень Р.А. Основы информатики и вычислительной техники. - 2-е изд. – М.: Просвещение, 1991. - 224 с.

9. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. - СПб.: БХВ-Петербург, 2002. - 608 с.

10. Романовская, Л.М. Программирование в среде Си. - М.: Финансы и статистика, 1992. - 350 с.

11. Butz A. R. Iterative algorithm for Hilbert’s space filling curve // IEEE Trans. On Computers. - 1971. - №20. - pp. 424-442.

12. Вирт Н. Алгоритмы и структуры данных. - 2-е изд. - М.: Книга по требованию, 2010. - 192 с.

13. Accelerated Parallel Processing OpenCL Programming Guide // developer.amd.com URL: developer.amd.com/wordpress/media/2013/07/AMD_Accelerated_Parallel_Processing_OpenCL_Programming_Guide-rev-2.7.pdf (дата обращения: 28.10.2015).

References

1. Gulyakovich G. N., Severtsev V.N., Shurchkov I.O. Inženernyj vestnik Dona (Rus). 2012. №2. URL: ivdon.ru/ru/magazine/archive/n2y2012/790

2. Voevodin V.V., Voevodin Vl. V. Otkrytye sistemy. 2013. №9. pp. 12-15.

3. Ponomareva E.I. Inženernyj vestnik Dona (Rus). 2012. №1. URL: ivdon.ru/ru/magazine/archive/n1y2012/628

4. Gervich L.R., Shteynberg B.Ya. Otkrytye sistemy. 2013. №8. pp. 26-29.

5. Warren G.S.-ml. Algoritmicheskie tryuki dlya programmistov [Algorithmic tricks for programmers]. 2-e izd. M.: Vil'yams, 2013. 512 p.

6. Aleksandrov P.S. Vvedenie v obshchuyu teoriyu mnozhestv i funktsiy [Introduction to general theory of sets and functions]. M.: Ogiz-Gostekhizdat, 1948. 413 p.

7. Krivye Gil'berta i Serpinskogo, ili snova rekursiya. 1september.ru URL: inf.1september.ru/1999/art/zlat1.htm (Date of access: 28.10.2015).

8. Kushnirenko A.G., Lebedev G.V., Skvoren' R.A. Osnovy informatiki i vychislitel'noy tekhniki [Basics of computer science]. 2-e izd. M.: Prosveshchenie, 1991. 224 p.

9. Voevodin V. V., Voevodin Vl. V. Parallel'nye vychisleniya [Parallel computations]. SPb.: BKhV-Peterburg, 2002. 608 p.

10. Romanovskaya, L.M. Programmirovanie v srede Si [C language programming]. M.: Finansy i statistika, 1992. 350 p.

11. Butz A. R. IEEE Trans. On Computers. 1971. №20. pp. 424-442.

12. Wirth N. Algoritmy i struktury dannykh [Algorithms and data structures]. 2-e izd. M.: Kniga po trebovaniyu, 2010. 192 p.

13. Accelerated Parallel Processing OpenCL Programming Guide. developer.amd.com.URL:developer.amd.com/wordpress/media/2013/07/AMD_Accelerated_Parallel_Processing_OpenCL_Programming_Guide-rev-2.7.pdf (request date: 28.10.2015).