ing for the Visualization of Functional Brain Images. – URL: http://www.vis.uni-stuttgart.de/ger/research/pub/pub2006/simvis06-roessler.pdf.

4. Flexible GPU-Based Multi-Volume Ray-Casting / R. Brecheisen [et al.] // Technical University of Eindhoven. – URL: http://www.yp. wtb.tue.nl/pdfs/9881.pdf

5. Beyer Johanna. GPU-based Multi-Volume Rendering of Complex Data in Neuroscience and Neurosurgery. – Dissertation. – Vienna University of Technology, okt. 2009. – 131 p.

6. The scientific computing and imagine institute at the university of Utah. – URL: http://www.sci.utah.edu/download/ IV3DData.html.

Р.К. Газизов, И.Р. Фатхулисламов

Уфимский государственный авиационный технический университет

GPU И MATLAB ДЛЯ РЕШЕНИЯ ЗАДАЧИ АВТОМАТИЧЕСКОЙ ГЕОГРАФИЧЕСКОЙ ПРИВЯЗКИ СПУТНИКОВЫХ ИЗОБРАЖЕНИЙ С ПИКСЕЛЬНОЙ ТОЧНОСТЬЮ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

В связи с резким увеличением количества графических карт для персональных компьютеров и рабочих станций графический процессор (GPU) стал платформой для параллельных вычислений, доступной для широкого круга пользователей. Учитывая его высокую пропускную способность памяти и высокую вычислительную производительность, GPU все чаще стали использовать для расчетов общего назначения (GPGPU). Множество удачных примеров использования GPU для реализации различных задач показаны в работе [1].

На сегодняшний день задача автоматической географической привязки спутниковых изображений является актуальной, так как в последнее время наблюдается устойчивая тенденция к увеличению объема получаемой информации из космоса и решение данной задачи требует значительного времени.

На первой стадии процесса получения продуктов обработки спутниковых данных выполняется первичная обработка цифровых изображений. Обязательным этапом обработки полученных данных является географическая привязка спутниковых изображений. От точности решения задачи географической привязки спутниковых изображений зависит решение множества других задач, например, коррекция геометрических искажений, синтез разнородных спутниковых изображений, оценка временной изменчивости ТПО, построение карт морских течений, определение скорости передвижения облачных масс, локализация пожаров и т.д. Постоянный рост числа запускаемых спутников, появление новых тематических задач делает необходимым, чтобы процедура географической привязки выполнялась автоматически и с высокой скоростью.

Решение задачи точной географической привязки спутниковых изображений невозможно без вычисления контрольных точек на изображениях. В данной статье рассмотрена параллельная реализация алгоритма Харриса [2] на языке программирования Matlab с использованием GPU.

Некоторые работы по использованию GPU в Matlab можно найти в [3]. В данной работе был использован коммерческий продукт Jacket [4]. Это набор библиотек для MATLAB, который позволяет использовать стандартный код MATLAB для работы на NVIDIA CUDA при поддержке GPU.

В наших экспериментах мы использовали ПК следующей конфигурации: Intel Core Duo E4600@2.4GHz, 8Gb RAM, NVIDIA GTX 280. Ниже показаны аппаратные особенности NVIDIA GTX 280:

Пиковая вычислительная производительность (TFlops)	0,62
Полоса пропускания памяти (ГБ/с)	141,7
Stream процессоры	240
Частота ядра (МГц)	602
Частота памяти (МГц)	1107
Объем памяти	1G
Общая память	16kB на SM

Во всех наших экспериментах при вычислении процессорного времени использовали соответствующие функции MATLAB. Результаты реализации алгоритмов на GPU сверены с результатами, которые были получены в MATLAB при расчетах на процессоре. Была произведена оптимизация исходного кода под использование данного GPU. Все изображения, которые были использованы в наших тестах, получены со спутников NOAA/AVHRR.

Одной из функций при использовании алгоритма Харриса является функция $conv2$, которая производит свертку исходного изображения и фильтра. Функция принимает в качестве входных параметров исходное изображение A и фильтр B, на выходе – изображение C. Формула для расчета пикселя $C_{m,n}$ выглядит следующим образом: $C[m,n] = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} B[j,k] A[m-j, n-k]$, где J и K – ширина и высота фильтра соответственно. Если фильтр соизмерим с исходным изображением, то мы можем использовать быстрое преобразование Фурье (БПФ), которое доступно в CUDA SDK. Однако в нашем случае размер фильтра 3×3 пикселя и применять БПФ нецелесообразно.

Учитывая небольшой размер регистровой памяти, мы хранили исходный фильтр в общей памяти, а не выделяли 9 регистров для каждого потока. Для хранения исходного изображения использовали текстурную память. Исходное изображение было разделено на колонки, равные количеству потоков. Каждый поток обрабатывал свою колонку.

Пропускная способность ядра 352 GFLOPS достигается при разрешении изображения 4096×4096 пикселей. Общее ускорение данного метода в 3,2 выше, чем при расчетах на CPU. Данный подход реализации функции $conv2$ на CUDA быстрее, чем поставляемая с коммерческим пакетом Jacket. Даже если не учитывать время на передачу данных между CPU GPU, двумерная свертка с ядром 3×3 пикселя на изображении 4096×4096 пикселей с использованием пакета Jacket занимает 0,114 с, в то время как оптимизированный код на CUDA и MEX занимает 0,091 с (учитывая время на передачу между

CPU и GPU). Как видно из рис. 1, используемый при расчетах CPU дает существенный прирост производительности.

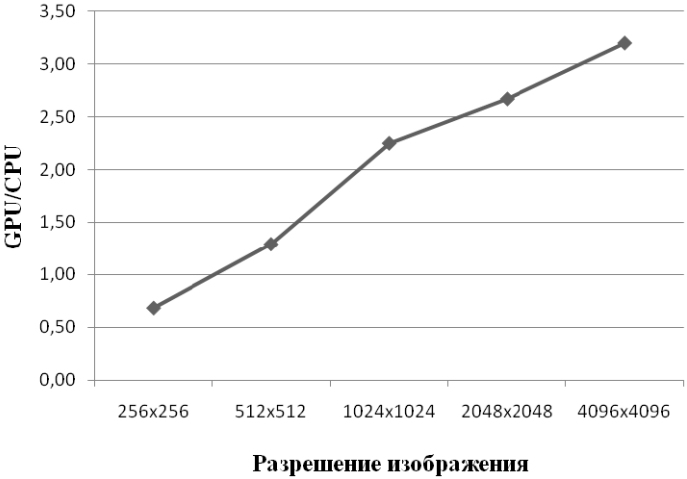

Рис. 1. Зависимость отношения ускорений GPU/CPU от разрешения изображения (функция *conv*2)

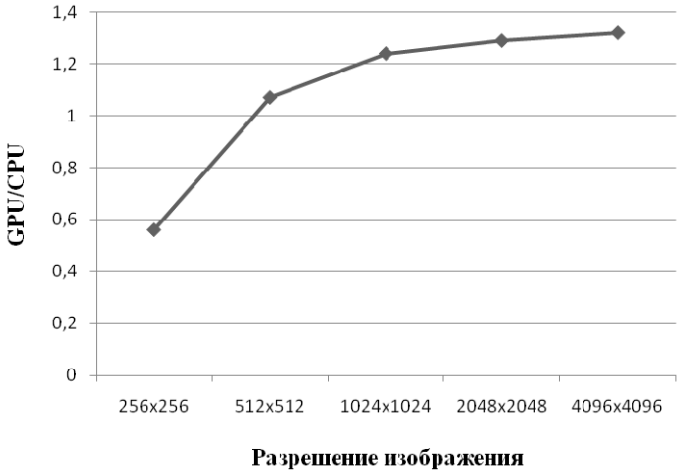

Рис. 2. Зависимость отношения ускорений GPU/CPU от разрешения изображения (алгоритм Харриса)

Распараллелив функцию *conv*2, общее ускорение алгоритма Харриса можно видеть на рис. 2. Как видно из полученных результатов, распараллелив функцию *conv*2 можно добиться ускорения на изображениях с разрешением выше, чем 512×512 пикселей.

Общий алгоритм автоматической географической привязки спутниковых изображений с пиксельной точностью был рассмотрен авторами ранее [5]. В данной статье выполнена параллельная реализация алгоритма Харриса с использованием GPU. Алгоритм был реализован с использованием среды программирования Matlab и библиотеки Jacket. Была произведена оптимизация исходного кода под использование данного GPU. Как видно из полученных результатов, оптимизация алгоритма под GPU дает прирост производительности на 20 %. Проведенные вычислительные эксперименты позволяют говорить об эффективности использования GPU для решения задачи поиска контрольных точек на изображениях. В дальнейшем планируется реализовать параллельный алгоритм на GPU других функций алгоритма и проверить алгоритм на различных графических картах.

Список литературы

1. GPU Computing / J.D. Owens [et al.]. Proceedings of the IEEE, 2008.

2. Harris C. and Stephens M. A combined corner and edge detector. In Proceedings, 4th Alvey Vision Conference. – Manchester, 1988. – P. 147–151.

3. http://developer.NVIDIA.com/object/MATLAB_cuda.html

4. http://www.accelereyes.com/products

5. Газизов Р.К., Фатхулисламов И.Р. Автоматическая географическая привязка спутниковых изображений с пиксельной в задаче минимизации объема хранимой информации // Перспективные информационные технологии для авиации и космоса: тр. междунар. конф. с элементами научной школы для молодежи. – Самара, 2010. – С. 748–752.