Быстрая иерархическая кластеризация мультиспектральных изображений на графических процессорах NVIDIA

С.А. Рылов[1], И.А. Пестунов[1]

[1]Институт вычислительных технологий СО РАН, проспект Академика Лаврентьева, 6, Новосибирск, 630090, Россия

Аннотация. В работе рассматривается реализация иерархического сеточного алгоритма кластеризации HCA на графических процессорах NVIDIA с использованием технологии CUDA, что позволило на порядок сократить время обработки мультиспектральных изображений. Представлены результаты экспериментальных исследований на модельных данных и изображениях, подтверждающие эффективность рассматриваемого алгоритма кластеризации и его параллельной реализации.

1. Введение

При решении целого ряда прикладных задач возникает необходимость кластеризации больших массивов данных при отсутствии каких-либо априорных сведений об искомых классах. Например, в задачах обработки мультиспектральных спутниковых изображений, содержащих десятки миллионов пикселей. В этих условиях целесообразно использовать вычислительно эффективные непараметрические алгоритмы. Этими сложно совместимыми свойствами обладают алгоритмы кластеризации, основанные на сеточном (grid-based) подходе [1], при котором происходит переход от обработки отдельных объектов к элементам сеточной структуры (клеткам), формируемой в пространстве признаков.

В работе [2] авторами были предложены иерархические сеточные алгоритмы кластеризации HCA и HECA, которые, в отличие от известных иерархических алгоритмов [3], позволяют разделять пересекающиеся кластеры.

С другой стороны, в настоящее время большинство настольных персональных компьютеров оснащено современными видеокартами, производительность которых в последние годы стремительно растет. На фоне этого активно развиваются технологии вычислений общего назначения на графических процессорах (ГП), позволяющие использовать их для решения вычислительно трудоемких задач [4].

В данной работе рассматривается реализация иерархического сеточного алгоритма кластеризации HCA на графических процессорах с использованием технологии CUDA, позволившая на порядок сократить время обработки мультиспектральных изображений. Достигнутое быстродействие делает возможным обработку видео потока высокого разрешения в реальном времени. Кроме того, представлены экспериментальные исследования алгоритма HCA на модельных данных и изображениях, подтверждающие высокое качество его работы.

2. Сеточный иерархический алгоритм кластеризации НСА

В данном разделе представлено краткое описание алгоритма кластеризации НСА.

Пусть множество классифицируемых объектов X состоит из векторов, лежащих в пространстве признаков R^d: $X = \{x_i = (x_i^1,\ldots, x_i^d) \in R^d,\ i = 1,\ldots,N\}$. Векторы x_i лежат в прямоугольном гиперпараллелепипеде $\Omega = [l^1, r^1]\times\ldots\times[l^d, r^d]$, где $l^j = min\ x_i^j,\ r^j = max\ x_i^j,\ x_i \in X$. Сеточная структура определяется как разбиение пространства признаков гиперплоскостями: $x^j = (r^j - l^j) \cdot i / m + l^j,\ i = 0,\ldots,m$, m – число разбиений Ω по каждой размерности. Минимальным элементом этой структуры является клетка (замкнутый прямоугольный гиперпараллелепипед, ограниченный гиперплоскостями). Введем общую нумерацию клеток (последовательно от одного слоя клеток к другому). Клетки B_i и B_j $(i \neq j)$ являются смежными, если их пересечение не пусто. Множество смежных с B клеток обозначим через A_B. Под плотностью клетки D_B будем понимать количество элементов множества X, попавших в клетку B. Клетку B будем считать непустой, если $D_B > 0$.

Непустая клетка B_i непосредственно связна с непустой клеткой B_j $(B_i \to B_j)$, если B_j – максимальная по номеру клетка, удовлетворяющая условиям: $B_j \in A_{Bi}$, и $\forall\ B \in A_{Bi}$: $D_{Bj} \geq D_B$. Непустые смежные клетки B_i и B_j непосредственно связны $(B_i \leftrightarrow B_j)$, если $B_i \to B_j$ или $B_j \to B_i$. Непустые клетки B_i и B_j связны $(B_i \sim B_j)$, если существуют $k_1,\ldots,k_l$ такие, что $k_1 = i,\ k_l = j$ и для всех $p = 1,\ldots,l-1$ выполнено $B_{kp} \leftrightarrow B_{kp+1}$. Введенное отношение связности порождает разбиение множества непустых клеток на компоненты связности $\{G_1,\ldots,G_S\}$. Под компонентой связности будем понимать максимальное множество попарно связных клеток. Представителем компоненты связности G назовем максимальную по номеру клетку $Y(G)$, удовлетворяющую условию $Y(G) = arg\ max_{B\in G}\ D_B$.

В алгоритме НСА в качестве элементов иерархии выступают не сами элементы исходных данных, а введенные выше компоненты связности. Благодаря использованию сеточной структуры число получаемых компонент связности мало относительно объема исходных данных, поэтому при построении иерархии не требуется больших вычислительных затрат.

Компоненты G_i и G_j считаются смежными, если существуют такие смежные клетки B_i и B_j, что $B_i \in G_i$ и $B_j \in G_j$. Расстояние между парой смежных компонент связности G_i и G_j определяется по формуле

$$h_{ij} = \min_{P_{ij}\in\Re_{ij}} \left[1 - \min_{B_{k_t}\in P_{ij}} D_{B_{k_t}} / \min(D_{Y_i}, D_{Y_j})\right],$$

где $\Re_{ij} = \{P_{ij}\}$ – множество всех цепей между представителями компонент связности $P_{ij} = \langle\ Y_i = B_{k_1},\ldots, B_{k_t}, B_{k_{t+1}},\ldots, B_{k_l} = Y_j\ \rangle$ таких, что для всех $t = 1,\ldots,l-1$: 1) $B_{k_t} \in G_i \cup G_j$; 2) $B_{k_t}, B_{k_{t+1}}$ – смежные клетки.

Матрица расстояний между компонентами связности $\{\hat{h}_{ij}\}$ строится на основе расстояний между смежными компонентами связности $\{h_{ij}\}$ следующим образом. Пусть $\Theta_{ij} = \{Q_{ij}\}$ – множество всех цепей из компонент связности $Q_{ij} = \langle\ G_i = G_{k_1},\ldots, G_{k_t}, G_{k_{t+1}},\ldots, G_{k_l} = G_j\ \rangle$ таких, что для всех $t = 1,\ldots,l-1$ компоненты $G_{k_t}, G_{k_{t+1}}$ смежные. Тогда расстояние между произвольными компонентами связности G_i и G_j будут вычисляться по формуле

$$\hat{h}_{ij} = \min_{Q_{ij}\in\Theta_{ij}} \left[\max_t h_{k_t,k_{t+1}}\right].$$

В случае если множество Θ_{ij} пусто, то полагаем $\hat{h}_{ij} = 1$.

Введенное расстояние на основе оценки плотности распределения данных позволяет избежать присущих иерархическим методам проблем с разделением пересекающихся классов. Отношение $\hat{h}_{ij}$ является ультраметрикой [5], т.е. метрикой, для которой выполняется усиленное неравенство треугольника: $\hat{h}_{ij} \leq max\ (\hat{h}_{ik}, \hat{h}_{kj}),\ \forall\ i, j, k$. Известно, что существует однозначное соответствие между матрицами расстояний со свойством ультраметрики и дендрограммами [6], то есть такие матрицы описывают иерархические разбиения.

Процедура получения ультраметрики $\{\hat{h}_{ij}\}$ из матрицы расстояний смежных объектов $\{h_{ij}\}$ в литературе известна как операция минимального транзитивного замыкания (min-transitive

closure) [7]. Для ее реализации, как правило, применяются вычислительно трудоемкие алгоритмы со сложностью $O(n^3)$ [6-8] (для матрицы расстояний размера $n \times n$). Однако оказалось [2], что операцию минимального транзитивного замыкания возможно выполнить с помощью алгоритма построения дендрограммы методом ближайшего соседа (SLINK), который можно реализовать с вычислительной сложностью $O(n^2)$. С учетом вышесказанного алгоритм HCA(m) можно записать в виде последовательности четырех основных шагов.

1. Формирование сеточной структуры. Для каждого вектора данных $x_i \in X$ определяется содержащая его клетка, и вычисляются плотности всех клеток.
2. Выделение компонент связности $\{G_1,\ldots,G_S\}$ и их клеток представителей $Y(G_1),\ldots, Y(G_S)$.
3. Формирование матрицы расстояний $\{h_{ij}\}$ между смежными компонентами связности в соответствии с определением.
4. Построение дендрограммы методом ближайшего соседа по матрице расстояний $\{h_{ij}\}$.

Алгоритм кластеризации HCA обладает линейной вычислительной сложностью от N (благодаря переходу к обработке сеточной структуры) и позволяет выделять иерархическую структуру данных, при этом он способен эффективно разделять пересекающиеся в пространстве признаков кластеры. Несмотря на экспоненциальную зависимость сложности от размерности данных d алгоритм эффективен для размерностей $d \leq 6$.

3. Реализация алгоритма кластеризации HCA с использованием архитектуры параллельных вычислений CUDA

Современные ГП содержат тысячи ядер, сгруппированных в блоки под управлением мультипроцессоров. Ядра одного блока выполняют одинаковый набор инструкций, но на различных элементах данных. Каждое ядро имеет небольшое число регистров, а также быстрый доступ к ограниченному объему разделяемой памяти внутри своего блока (управляемый кэш). Кроме того, всем потокам (исполняемым на отдельных ядрах) доступен большой объем глобальной памяти видеокарты, однако время случайного доступа к нему занимает сотни циклов. Синхронизация потоков во время выполнения возможна только в рамках своего блока.

При реализации алгоритма с помощью технологии CUDA мы ориентировались на архитектуру графических процессоров Kepler и использовали ее возможности, которые не доступны на более ранних версиях: быстрые атомарные операции с глобальной памятью, shuffle-инструкции и другие. Для достижения максимальной эффективности на ГП были реализованы варианты алгоритма для фиксированных размерностей данных: 3 и 4 (для ЦПУ алгоритм реализован для произвольной размерности вплоть до 8).

В параллельной реализации алгоритма HCA выделяются шесть основных этапов.

На первом этапе осуществляется вычисление плотностей клеток в формируемой сетке. Этот процесс можно рассматривать как вычисление многомерной гистограммы. Элементы массива гистограммы заполняются в глобальной памяти видеокарты с помощью атомарной операции инкремента. Распространенные подходы построения гистограммы на ГП, использующие разделяемую память, не подходят для обработки многомерной гистограммы из-за сильно ограниченного размера разделяемой памяти (не более 48 КБ).

На втором этапе для каждой клетки выявляется смежная с ней клетка с максимальной плотностью. При реализации в рамках CUDA каждая клетка обрабатывается одним потоком, просматривающим соседние с ней клетки. Для увеличения производительности был использован ряд оптимизаций.

- Использование разделяемой памяти для обмена результатами. Потоки каждого блока выровнены в линию вдоль оси Z, и каждый поток просматривает только соседние клетки с фиксированной координатой z. Затем происходит обмен промежуточными результатами между соседними потоками с помощью разделяемой памяти.
- Исключение повторного считывания элементов. Каждый поток последовательно обрабатывает группу элементов, смещаясь по оси Y. При этом используются уже найденные промежуточные максимумы для каждого пройденного слоя с фиксированной координатой y, что позволяет просматривать только 1/3 часть элементов, требуемых для определения максимума в новой позиции.

На втором этапе также определяются клетки представители компонент связности.

На третьем этапе выполняется непосредственное выделение компонент связности. Каждый поток обрабатывает одну клетку, последовательно переходя по ссылкам к смежным клеткам с максимальной плотностью вплоть до достижения одного из представителей компонент связности, к которой затем относится пройденная цепочка клеток.

На четвертом этапе формируется матрица расстояний между смежными компонентами связности согласно определению. По построению наименьшая плотность между смежными компонентами достигается на их граничных клетках. Поэтому в ходе параллельной обработки пар смежных клеток, в случае, если клетки принадлежат различным компонентам, вычисляется значение перепада плотности между соответствующими компонентами по рассматриваемым клеткам. После чего расстояние между компонентами обновляется в глобальной памяти с помощью атомарной операции минимума.

Пятый этап алгоритма состоит в построении дендрограммы с помощью применения метода одиночной связи (SLINK) к матрице расстояний между смежными компонентами связности. Алгоритм SLINK для матрицы размера $n \times n$ состоит из (n-1) итерации, на каждой из которых объединяются два ближайших кластера, при этом расстояние до нового кластера определяется как минимальное из расстояний до объединяемых кластеров по свойству SANN (same agglomerative nearest neighbor). В работе [9] изложен алгоритм на основе использования массива частичных минимумов (содержащих индексы и значения минимальных элементов в каждой строке матрицы расстояний), позволяющий находить минимальный элемент матрицы за $O(n)$ операций. Свойство SANN позволяет обновлять матрицу расстояний и массив частичных минимумов за $O(n)$. Таким образом, достигается общая вычислительная сложность метода одиночной связи $O(n^2)$.

Каждая последующая итерация (уровень дендрограммы) непосредственно зависит от предыдущей, что затрудняет возможность эффективного распараллеливания. Известные методы параллельной реализации SLINK, как правило, делят данные на пересекающиеся части, проводят независимое построение дендрограмм для каждой из частей и затем объединяют частичные результаты [9-11]. Однако такой подход затруднителен для реализации на графических процессорах, рассчитанных на массовые однотипные вычисления.

При реализации на ГП распараллеливание производилось за счет параллельного пересчета расстояний до нового кластера (получающегося в результате последнего объединения) и эффективного нахождения минимального элемента в массиве, реализованного на основе shuffle-инструкций, позволяющих обмениваться значениями регистров между соседними потоками без явной синхронизации и использования разделяемой памяти [12]. Во избежание проблем с синхронизацией выполняемых блоков и обмена информацией между ними, весь процесс реализуется на одном блоке потоков. При этом в разделяемой памяти блока размещается массив частичных минимумов. Ограничение размера разделяемой памяти (48 KB) при таком подходе ограничивает размер обрабатываемой матрицы до 6100×6100, что является достаточным, так как количество компонент связности на практике обычно не превосходит 2000 (для мультиспектральных спутниковых изображений).

В литературе по реализации алгоритма SLINK на ГП было найдено исследование [13], в котором используется схожий подход к распараллеливанию на основе массива частичных минимумов. В своей работе авторы обрабатывали матрицы размером от 4096×4096 до 16384×16384 и получили коэффициент ускорения SLINK на ГП от 0.8 до 2.1. При этом необходимо отметить, что ими использовалась видеокарта Nvidia Tesla C870.

Реализованный в данной работе метод SLINK на ГП превосходит по скорости работы аналогичную реализацию на ЦПУ начиная с n=500 и дает ускорение в 3 раза при n=1000 и более чем в 7 раз при n>3000. Несмотря на проигрыш в скорости работы метода SLINK на ГП для меньших матриц расстояний, время их обработки не превосходит 1 мс. Поэтому имеет смысл также выполнять данный этап вычислений на видеокарте, чтобы не прерывать процесс дополнительными копированиями данных между ЦПУ и ГП.

На заключительном этапе алгоритма элементы исходных данных распределяются по кластерам, к которым отнесены содержащие их клетки.

4. Результаты экспериментальных исследований

Для сравнения качества работы алгоритма HCA с другими алгоритмами использовался открытый программный пакет ELKI [14], включающий в себя такие известные алгоритмы кластеризации, как K-means, EM, DBSCAN, OPTICS, DeLiClu, SLINK, Mean-shift.

На рисунке 1*а* представлены модельные данные, состоящие из восьми нормально распределенных классов [15]. Классы сгруппированы в три изолированные группы, в одной из которых имеются значительные пересечения, что существенно затрудняет их разделение. Алгоритм HCA позволил выделить все восемь классов (рисунок 1*б*) с точностью 97.98%, а также позволил выявить иерархическую структуру данных (рисунок 1*б-г*). В данном случае точность кластеризации определяется, как процент правильно расклассифицированных элементов. Для этого каждому классу эталонного разбиения ставится в соответствие кластер (не более одного), в который попало наибольшее число элементов из этого класса.

Алгоритмы кластеризации K-средних и EM, предназначенные для разделения нормально распределенных классов, способны выделить все 8 классов только при удачной инициализации центров. В то же время плотностные алгоритмы (DBSCAN, OPTICS, DeLiClu), а также иерархические методы (ближайшего соседа, дальнего соседа, взвешенной средней связи) не способны разделить пересекающуюся группу из трех кластеров. В результате их применения выделяется не более шести кластеров (в зависимости от значений выбранных параметров).

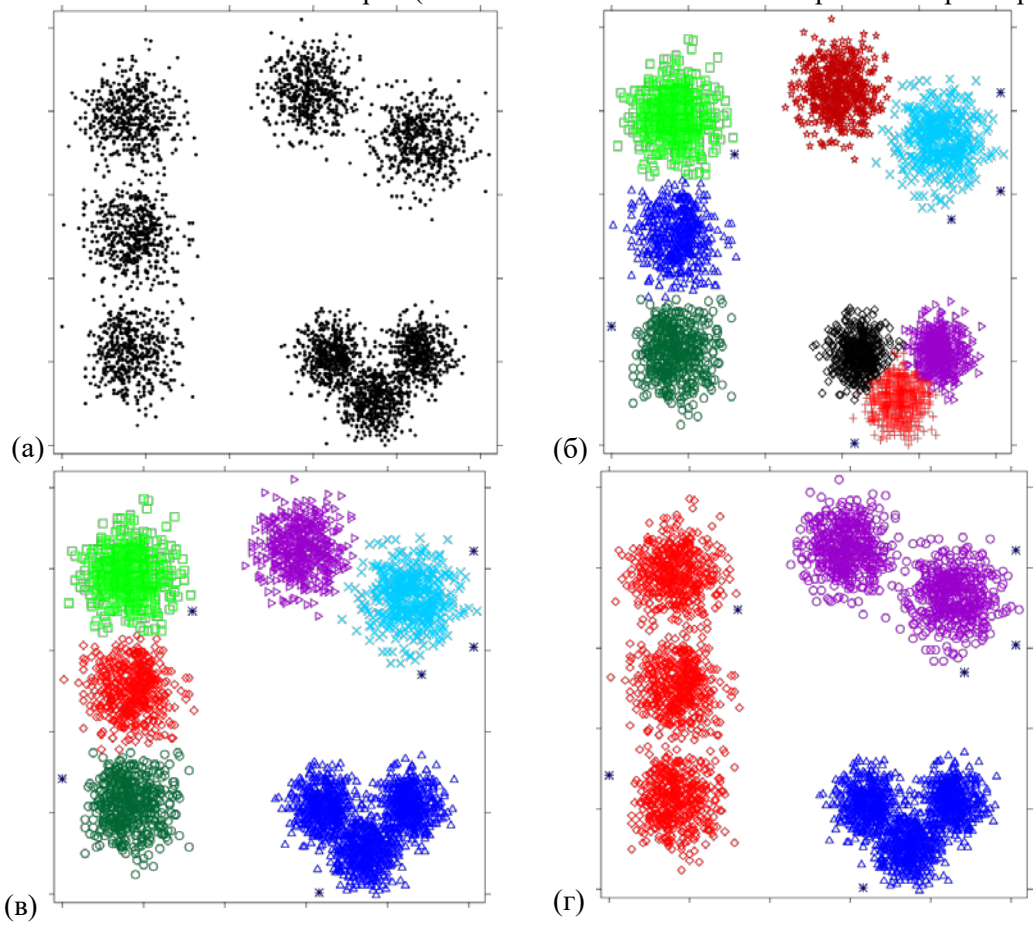

Рисунок 1. Модельные данные (а); результаты кластеризации алгоритмом HCA на различных уровнях иерархии (б-г).

Вторая рассматриваемая модель, представленная на рисунке 2*а*, состоит из пяти классов, различающихся по форме, размеру и плотности, включая кольцевые и нормально распределенные [15]. Алгоритм HCA позволил выделить все классы с точностью 99.44% (рисунок 2*б*). Однако ни один алгоритм из пакета ELKI не смог правильно выделить все пять

классов. Наилучшие результаты показал плотностной иерархический алгоритм OPTICS, обеспечив точность 79.7% (рисунок 2*в*), а также иерархический алгоритм ближайшего соседа (SLINK) с точностью 72.72% (рисунок 2*г*).

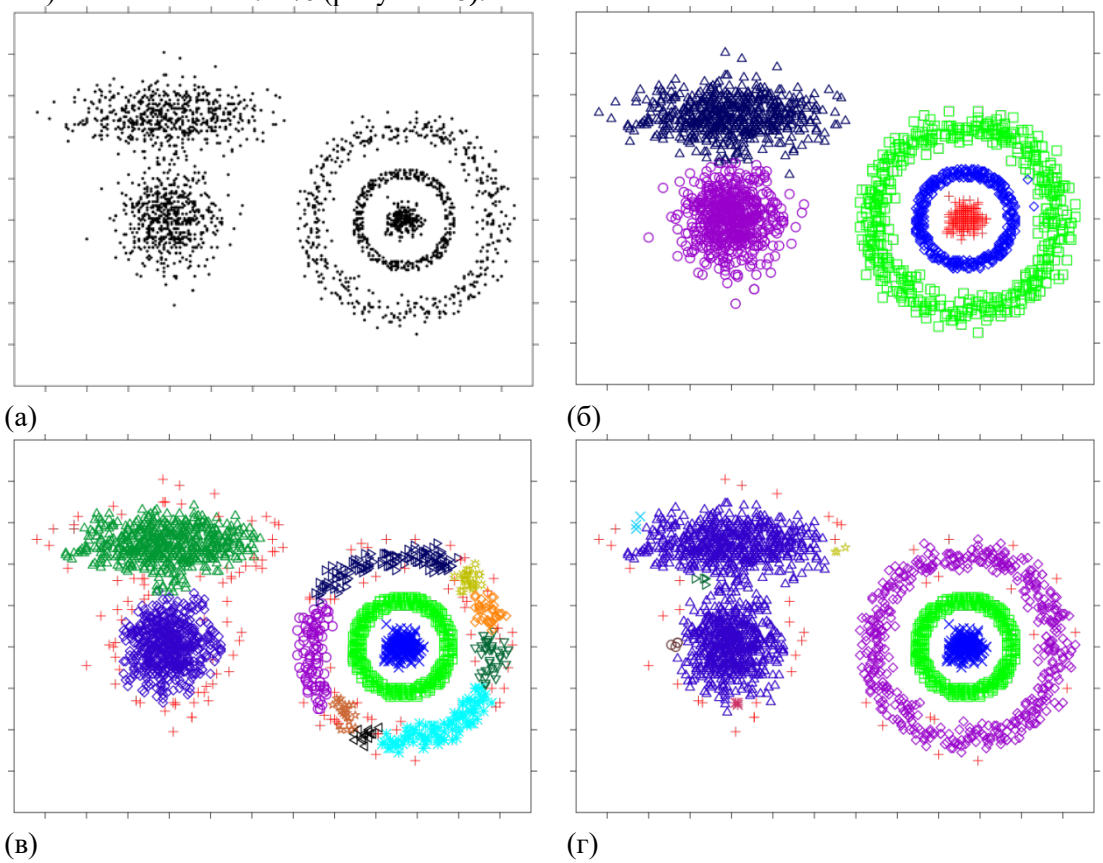

Рисунок 2. Модель из пяти классов, различающихся по форме, размеру и плотности (а); результат кластеризации алгоритмами HCA (б), OPTICS (в) и SLINK (г).

Вычислительная эффективность алгоритма HCA позволяет проводить обработку мультиспектральных изображений. На рисунке 3 приведен пример кластеризации снимка, полученного со спутника WorldView-2, при различных уровнях иерархии. При кластеризации использовалось четыре канала: 1, 2, 4, 7. Размер изображения составляет 2048 × 2048 пикселей.

В таблице 2 приведено сравнение времени работы алгоритма HCA при исполнении на ГП (GeForce GXT 770), а также на одном и четырех ядрах ЦП (Intel Core i7 960, 3.2 ГГц) при обработке RGB-изображений и четырехканальных спутниковых снимков разного размера. Параллельная версия алгоритма для ЦП реализована с помощью технологии OpenMP. Параметр сетки m, который непосредственно влияет на подробность получаемой сегментации, задавался равным 32 (рекомендуемое начальное значение для изображений). Результаты показывают, что полученное ускорение при использовании видеокарты составляет порядка 15 раз для цветных изображений и порядка 10 раз для четырехканальных.

Таблица 2. Время работы алгоритма HCA на изображениях (указано в мс).

	Цветные изображения			**Спутниковые снимки**		
Размер (млн. пикселей)	5	12.4	115	4	24	100
ЦП, 1 ядро	47	95	722	82	290	954
ЦП, 4 ядра	36	70	718	62	240	767
ГП	3	5	37	7	23	64

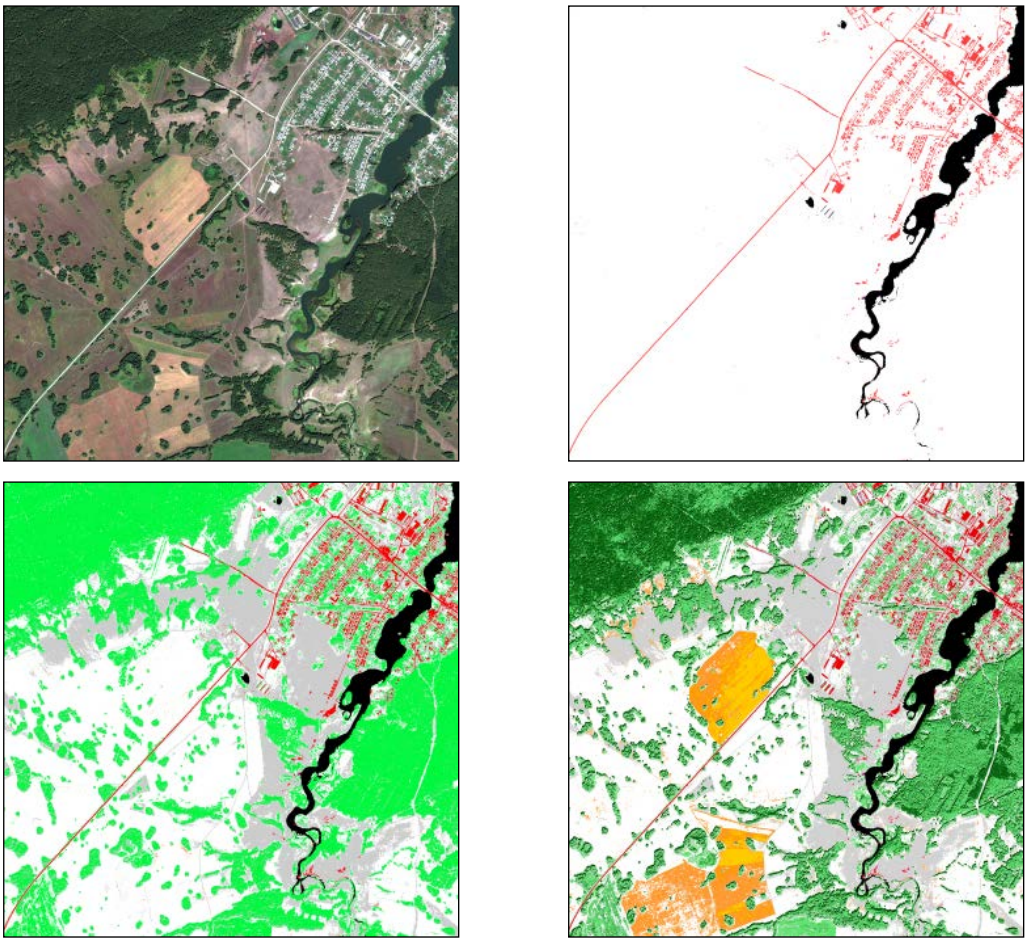

Рисунок 3. Спутниковое изображение WorldView-2 (RGB-композит, каналы 5, 3, 2) и результат кластеризации алгоритмом HCA на разных уровнях иерархии.

5. Заключение

В работе представлена реализация иерархического сеточного алгоритма кластеризации HCA на графических процессорах NVIDIA для сегментации мультиспектральных изображений. Результаты экспериментальных исследований на модельных данных и изображениях показали, что рассматриваемый алгоритм обеспечивает получение качественных результатов кластеризации, а использование ГП позволяет сократить время обработки изображений в 10-15 раз по сравнению с ЦПУ. Достигнутое быстродействие делает возможным обработку видео потока высокого разрешения в реальном времени.

Предложенную параллельную реализацию возможно использовать и для ансамблевой версии алгоритма – HECA [2], который позволяет существенно упростить выбор параметра сетки *m*. В дальнейшем планируется реализовать на графических процессорах новый алгоритм кластеризации HCA-MS [16], который основан на алгоритме HCA, и в своей работе использует вычислительно трудоемкую процедуру среднего сдвига для уточнения границ кластеров. Мы ожидаем, что при этом время обработки спутниковых изображений сократится с нескольких минут до нескольких секунд.

Литература

[1] Ilango, M.R. A survey of grid based clustering algorithms / M.R. Ilango, V. Mohan // Intern. J. Eng. Sci. and Technology. – 2010. – Vol. 2(8). – P. 3441-3446.

[2] Пестунов, И.А. Иерархические алгоритмы кластеризации для сегментации мультиспектральных изображений / И.А. Пестунов, С.А. Рылов, В.Б. Бериков // Автометрия. – 2015. – Т. 51, № 4. – С. 12-22.
[3] Xu, R. Survey of clustering algorithms / R. Xu, D.I. Wunsch // IEEE Trans. on Neural Networks. – 2005. – Vol. 16(3). – P. 645-678.
[4] Боресков, А.В. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / А.В. Боресков, А.А. Харламов, Н.Д. Марковский, Д.Н. Микушин, Е.В. Мортиков, А.А. Мыльцев, Н.А. Сахарных, В.А. Фролов. – М.: Изд-во Московского университета, 2012. – 336 с.
[5] Leclerc, B. Description combinatoire des ultramétriques / B. Leclerc // Math. Sci. Humaines. – 1981. – Vol. 127(73). – P. 5-37.
[6] Mirzaei, A. A novel hierarchical-clustering-combination scheme based on fuzzy-similarity relations / A. Mirzaei, M. Rahmati // IEEE Tr. Fuzzy Syst. – 2010. – Vol. 18(1). – P. 27-39.
[7] Zheng, L. Hierarchical Ensemble Clustering / L. Zheng, T. Li, C. Ding // Proceedings of 2010 IEEE International Conference on Data Mining, IEEE. – 2010. – P. 1199-1204.
[8] Skiena, St.S. The Algorithm Design Manual / St.S. Skiena. – Springer, 2008. – 730 p.
[9] Olson, Cl.F. Parallel algorithms for hierarchical clustering / Cl.F. Olson // Parallel Computing. – 1995. – Vol. 21(8). – P. 1313-1325.
[10] Jin, C. DiSC: A Distributed Single-Linkage Hierarchical Clustering Algorithm using MapReduce / C. Jin, M.A. Patwary, A. Agrawal, W. Hendrix, W. Liao, A. Choudhary // Proceedings of the 4th International SC Workshop on Data Intensive Computing in the Clouds. – 2013. – Vol. 23. – P. 27.
[11] Hendrix, W. Parallel hierarchical clustering on shared memory platforms / W. Hendrix, M.A. Patwary, A. Agrawal, W. Liao, A. Choudhary // 19th Annual International Conference on High Performance Computing, HIPC. – 2012. – P. 1-9.
[12] Luitjens J. Faster Parallel Reductions on Kepler [Electronic resource]. – URL: http://devblogs.nvidia.com/parallelforall/faster-parallel-reductions-kepler (21.11.2017).
[13] Chang, D.-J. Hierarchical clustering with cuda/gpu / D.-J. Chang, M. Kantardzic, M. Ouyang // 22nd International Conference on Parallel and Distributed Computing and Communication Systems, ISCA PDCCS. – 2009. – P. 7-12.
[14] Schubert, E. A Framework for Clustering Uncertain Data / E. Schubert, A. Koos, T. Emrich, A. Züfle, Kl.A. Schmid, A. Zimek // Proc. VLDB Endowment. – 2015. – Vol. 8(12). – P. 1976-1979.
[15] Рылов, С.А. Модельные данные для кластеризации [Электронный ресурс]. – URL: https://drive.google.com/open?id=0ByK9GtU5ExExRnZwdFNmRHRWdFk (21.11.2017).
[16] Рылов, С.А. Непараметрический алгоритм кластеризации для сегментации изображений на основе комбинации сеточного подхода и процедуры среднего сдвига / С.А. Рылов // Сборник трудов Всероссийской конференции «Обработка пространственных данных в задачах мониторинга природных и антропогенных процессов». – Новосибирск, 2017. – С. 150-155.

Fast hierarchical clustering of multispectral images and its implementation on NVIDIA GPU

S.A. Rylov[1], I.A. Pestunov[1]

[1]Institute of computational technologies SB RAS, 6 acad. Lavrentiev ave., Novosibirsk, Russia, 630090

Abstract. The present work explores the parallel implementation of the hierarchical grid-based clustering algorithm HCA on NVIDIA GPU with CUDA technology, which substantially reduced the processing time of multispectral images. Provided experimental studies on model data and images confirm the efficiency of the HCA clustering algorithm and its parallel implementation.

Keywords: fast clustering, multispectral images, segmentation, hierarchical clustering, grid-based approach, parallel computing, GPU, GPGPU, CUDA, HCA.