М.П. Димашова

Нижегородский государственный университет им. Н.И. Лобачевского

РЕАЛИЗАЦИЯ АЛГОРИТМА СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ MEAN SHIFT НА GPU[1]

Сегментация изображения – это разбиение изображения на множество покрывающих его областей [1]. Сегментация изображений является одной из базовых задач компьютерного зрения, поскольку анализ изображения часто начинается с его декомпозиции на области, с которыми связана существенная для данной задачи информация.

Алгоритмы сегментации изображений можно разделить на два класса [2]: интерактивные – использующие пользовательские подсказки и автоматические, – не требующие участия пользователя (однако существуют и другие классификации [3]). Автоматические алгоритмы сегментации, в свою очередь, также делятся на два класса: 1) выделение областей изображения с заданными свойствами, специфичными для конкретной предметной области; 2) разбиение изображения на однородные области. Второй класс алгоритмов автоматической сегментации не использует априорную информацию об изображении, но к свойствам получаемых сегментов предъявляются некоторые требования, такие как однородность текстуры внутри сегмента и непохожесть текстур смежных областей, гладкость границ сегментов и т.п. [1]. Алгоритмы сегментации, разбивающие изображение на однородные области, являются наиболее универсальными, поскольку не ориентированы на определенную предметную область и находят широкое применение в компьютерном зрении при распознавании изображений (медицинских снимков [4], кад-

[1] Работа выполнена при финансовой поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839.

ров видео автоматизированной сборки деталей и т.п.), поиске стереосоответствий [5] и т.д. Одним из таких автоматических алгоритмов сегментации изображений является алгоритм Mean shift [6].

В данной работе выполнена программная реализация алгоритма сегментации Mean shift на GPU с целью последующего использования в системе анализа медицинских снимков и системе нахождения стереосоответствий. Выбор именно этого алгоритма сегментации среди прочих других [1] связан со следующими фактами:

1. Mean shift алгоритм сегментации имеет математическое обоснование (не является эвристическим).

2. Mean shift алгоритм, относясь к классу алгоритмов кластеризации, не требует задания количества кластеров, в отличие от других алгоритмов кластеризации, применяемых для сегментации изображений (kmeans [1] и т.д.). Это означает, что Mean shift-алгоритм определяет количество сегментов изображения в процессе своей работы.

3. Итеративная обработка пикселей изображения в Mean shift допускает распараллеливание на графических сопроцессорах, так как результат обработки одного пикселя на каждой итерации не зависит от результата обработки остальных пикселей. Большинство алгоритмов сегментации изображений являются вычислительно трудоемкими, но не каждый укладывается в модель массивно-параллельных вычислений.

4. О хорошем «качестве» алгоритма сегментации изображений Mean shift свидетельствуют многочисленные публикации, в которых Mean shift применяется при решении конкретных задач компьютерного зрения [4, 5,7]. Кроме того, стоит отметить, что задача разбиения изображения на однородные области является некорректно поставленной задачей, поскольку одно и то же изображение можно разбить на сегменты разными способами, и нет объективного критерия, позволяющего определить, какое из разбиений является «правильным». Выбор «лучшего» алгоритма зависит от решаемой задачи. Для сравнения

качества методов сегментации существуют базы изображений, для которых известна некоторая «эталонная» сегментация.

Основное внимание в данной работе уделено возможности ускорения алгоритма сегментации Mean shift с использованием графических сопроцессоров. В качестве CPU реализации Mean shift, послужившей основой для дальнейшего распараллеливания на GPU, была взята реализация из популярной библиотеки компьютерного зрения с открытым исходным кодом OpenCV (https://code.ros.org/gf/project/opencv). Вопрос качества реализованного в данной работе алгоритма не рассматривается, качество оценено лишь субъективно (визуально результат совпадает с результатом OpenCV).

Постановка задачи. Алгоритм сегментации Mean shift, как уже упоминалось, относится к классу алгоритмов кластеризации. Задача сегментации изображения формулируется как задача кластеризации.

Дано цветное изображение $I = \{(r,g,b)_i, i = 1,...n\}$ – будем для простоты рассматривать цветовое пространство RGB.

Необходимо разбить заданную выборку (изображение) на кластеры (сегменты) в пространстве признаков, представляющем собой объединение цветового пространства и пространства координат изображения $RGBXY$. В качестве меры близости экземпляров выборки может служить евклидово расстояние в пространстве признаков.

Краткое описание алгоритма. Основная идея Mean shift-метода [4] заключается в том, что по входным данным (изображению) можно построить ядерную оценку для плотности вероятности распределения данных в пространстве признаков $RGBXY$. Далее делается естественное предположение о том, что локальные максимумы плотности вероятности соответствуют центрам кластеров. Из необходимого условия локального экстремума определяется выражение для вектора сдвига $m(p)$ точки пространства признаков $p \in RGBXY$, применяя который к точке p итеративно получаем последовательность точек, схо-

дящуюся к локальному максимуму оценки плотности вероятности (т.е. к центру ближайшего кластера):

$$m(p) = \frac{\sum_{i=1}^{n} p_i g_i}{\sum_{i=1}^{n} g_i} - p, \qquad (1)$$

где $g_i = g(\| (p - p_i) / h \|^2)$, $g(v) = -k'(v)$, h – параметр сглаживания, $K(v) = ck(v)$ – ядро оценки плотности вероятности.

Например, ядро Епанечникова

$$K(v) = c \begin{cases} 1 - \| v \|^2, \| v \| \le 1 \\ 0, \textit{иначе.} \end{cases}$$

Для объединенного пространства $RGBXY$ предполагается, что

$$K((r,g,b,x,y)) = ck((r,g,b))k((x,y)).$$

В данной работе по аналогии с OpenCV используется

$$g_i = \begin{cases} 1, \| (r,g,b) - (r,g,b)_i \| \le h_{RGB} \ \& \ \| (x,y) - (x,y)_i \| \le h_{XY} \\ 0, \textit{иначе.} \end{cases};$$

где h_{RGB} и h_{XY} – параметры сглаживания в соответствующих пространствах.

Теперь можно представить алгоритм сегментации Mean shift. Дано изображение $I = \{(r,g,b)_i, i = 1,...n\}$ и параметры сглаживания h_{RGB} и h_{XY}.

1. Для каждого пикселя $p = (r,g,b;x,y)$ применить итеративно сдвиг

$$p^{(t)} = p^{(t-1)} + m(p^{(t-1)}), \qquad (2)$$

где t – номер итерации, $p^{(0)} = p$. Критерии останова: $t < T$ или $\| m(p^{(t)}) \| < \varepsilon$, ($T$, ε задаются пользователем).

2. Объединить пиксели, «пришедшие» в окрестность одного локального максимума плотности вероятности, в один сегмент. На этом этапе возможна фильтрация сегментов, имеющих маленькую площадь.

Реализация алгоритма. Алгоритм сегментации изображений Mean shift, как следует из описания, состоит из двух частей: итеративный сдвиг пикселей (1-й этап) и выделение связных компонентов (2-й этап).

Первый этап является основным и наиболее вычислительно трудоемким. Именно первый этап алгоритма укладывается в модель массивно-параллельных вычислений и реализован с помощью технологии CUDA на GPU. Входное изображение загружается в 4-канальную текстуру GPU. Остальные входные данные – параметры алгоритма – передаются через разделяемую память. Для хранения выходного изображения (в текущей реализации оно представляет собой изображение исходного размера, содержащее цвета, к которым сошлась процедура итеративного сдвига) используется глобальная память GPU. Обработка каждого пикселя (итеративный сдвиг) выполняется отдельным потоком, т.е. ядро программы представляет собой вычисления по формулам (2) и (1).

Постобработка по выделению связных компонентов выполнена известным рекурсивным алгоритмом Flood fill на CPU.

Результаты. В этом разделе приводится сравнение времени работы выполненной GPU реализации алгоритма Mean shift и его CPU реализации, взятой из библиотеки OpenCV (таблица). Время постобработки по выделению связных компонентов в представленные оценки не входит, поскольку в обеих реализациях она выполняется на CPU и составляет несущественную часть общего времени работы алгоритма.

Вычислительные эксперименты проводились на следующих процессорах:

- CPU – Intel Core i5 750, 2.66 GHz.
- GPU – NVidia GeForce GT 220 (48 ядер) и NVidia GeForce GTX 470 (Fermi, 448 ядер).

Отметим, что CPU реализация Mean shift – однопоточная, т.е. использовалось только одно из четырех ядер Intel Core i5.

Сравнение времени работы Mean shift на CPU и GPU

Размер изображения (пикс.)	Время работы (мс)					Ускорение	
	Intel Core i5	NVidia GT 220 (без i/o данных на GPU)	NVidia GT 220 (с i/o данных на GPU)	NVidia GTX 470 (без i/o данных на GPU)	NVidia GTX 470 (с i/o данных на GPU)	NVidia GT 220 (с i/o данных на GPU)	NVidia GTX 470 (с i/o данных на GPU)
300×225	6895,88	742,577	743,583	80,3896	81,0411	9,273	85,091
700×525	37685,9	3740,96	3744,84	385,637	387,937	10,063	97,144
1000×750	77237,9	8184,33	8190,35	788,693	792,476	9,430	97,464
2000×1500	304973	timed out	timed out	3106,63	3122,19	timed out	97,679
3072×2304	707879	timed out	timed out	7234,29	7264,26	timed out	97,446

Оценка времени работы алгоритма на GPU проводилась с учетом и без учета времени копирования данных с CPU на GPU и обратно. Для NVidia GT 220 не приведено время работы на больших изображениях, поскольку для этого графического сопроцессора существует ограничение на время работы ядра. Далее приведена диаграмма ускорения, полученного на GPU по сравнению с CPU (рис. 1), и диаграмма ускорения, приходящегося на одно ядро GPU (рис. 2).

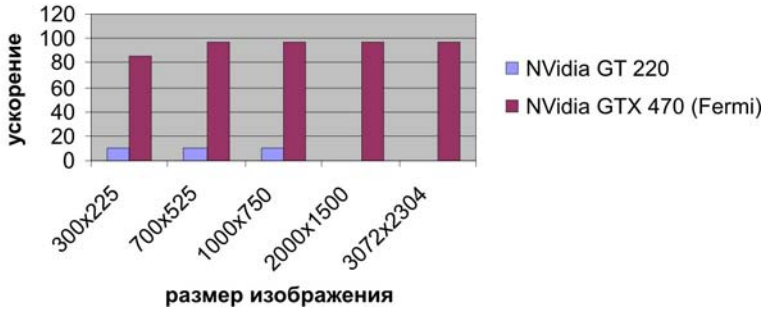

Рис. 1. Ускорение GPU реализации по сравнению с CPU

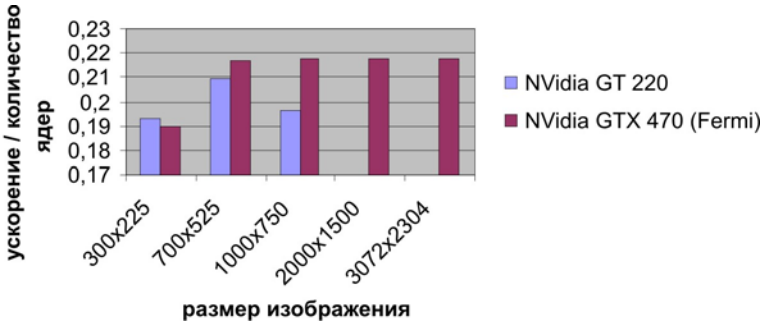

Рис. 2. Ускорение, приходящееся на одно ядро GPU

Ранее алгоритм сегментации изображений Mean shift уже реализовывался на GPU. В работе [6] пишут об ускорении до 134 раз на графическом сопроцессоре NVidia GTX 280 (240 ядер). Однако сравнивать ускорение, полученное в [6] и в данной работе не совсем корректно, поскольку в [6] не описан объем проводимых вычислений (неизвестны параметры алгоритма и его реализация).

Приведено сравнение времени работы последовательной реализации алгоритма сегментации изображений Mean shift, взятой из библиотеки компьютерного зрения OpenCV и выполненной GPU-реализации. Оценка времени работы проводилась на двух видах графических сопроцессоров, включая NVidia Fermi. Современные GPU позволили получить для алгоритма Mean shift ускорение до 98 раз.

Автор благодарен д.т.н. В.Е. Турлапову за обсуждение результатов работы.

Список литературы

1. Шапиро Л., Стокман Дж. Компьютерное зрение. – М.: Бином. Лаборатория знаний, 2006. – 752 с.

2. Баринова О., Вежневец А. Методы сегментации изображений: автоматическая сегментация. – URL: http://cgm.computergraphics. ru/content/view/147.

3. Zhang Y. Advances in Image And Video Segmentation. – USA: IRM Press., 2006.

4. Kim E., Wang W., Li H., Huang X. A parallel annealing methods for automatic color cervigram image segmentation. – URL: http://edwardkim.net/publications/edkim_miccaigpu2009.pdf.

5. Klaus A., Sormann M., Karner K. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. ICPR 2006. – URL: http://old.vrvis.at/2d3d/technology/stereomatching/images/segment-based-stereomatching.pdf.

6. Comaniciu D., Meer P. Mean shift: A robust approach towards feature space analysis. IEEE Trans. Pattern Analysis and Machine Intelligence, 24: 603–619, 2002.

7. Wang Z., Zheng Z. A region based stereo matching algorithm using cooperative optimization. CVPR, 2008. – URL: http://vision. middlebury.edu/stereo/eval/papers/CORegion.pdf.

Д.Н. Добряев, Н.В. Данилова

Нижегородский государственный университет им. Н.И. Лобачевского

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ГЛОБАЛЬНОГО ПОИСКА С ОБОБЩЁННОЙ ХАРАКТЕРИСТИКОЙ КВАДРАТИЧНОГО ТИПА

Быстрое развитие вычислительных средств, в частности появление возможности распараллеливания, позволяет решать оптимизационные проблемы [1, 2, 3]. Однако наряду с использованием новых высокопроизводительных технологий важное место занимает используемая методика поиска оптимума. Поэтому создание систем, которые помимо эффективного использования вычислительных ресурсов предлагают широкий спектр возможностей выбора метода в зависимости от поставленной задачи, является перспективным направлением на пути получения значимых результатов. А это, в свою очередь, требует создания новых алгоритмов, адаптированных к специфике решаемой задачи.