ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

ОБРАБОТКА СИГНАЛОВ, ИЗОБРАЖЕНИЙ, РЕЧИ, ТЕКСТА И РАСПОЗНАВАНИЕ ОБРАЗОВ

SIGNAL, IMAGE, SPEECH, TEXT PROCESSING AND PATTERN RECOGNITION

УДК 004.92

Поступила в редакцию 08.10.2018
Received 08.10.2018

Принята к публикации 20.02.2019
Accepted 20.02.2019

Алгоритм быстрого вычисления локальных гистограмм изображения на видеокарте

Ф. С. Троцкий, Б. А. Залесский✉

Объединенный институт проблем информатики
Национальной академии наук Беларуси, Минск, Беларусь
✉*E-mail: zalesky@newman.bas-net.by*

Аннотация. Рассматривается алгоритм параллельного вычисления гистограмм различных типов, в том числе яркости и ориентированного градиента, предназначенный для выполнения на видеокартах, которые поддерживают массивные параллельные вычисления. В настоящее время локальные гистограммы используются для решения задач обработки и распознавания изображений, однако их применение ограничено из-за большого времени вычисления для всех пикселов изображения. Одна из основных трудностей, возникающих при вычислении этих векторных признаков, – большое число конфликтов одновременного доступа к ячейкам видеопамяти, в которые записываются одинаковые значения характеристики. В предложенном алгоритме существенно уменьшено число конфликтов одновременного доступа, что позволило значительно уменьшить время его выполнения. Так, например, девятимерные векторы локальных гистограмм ориентированного градиента для всех 256×256 окон изображения размера HD вычисляются на видеокарте GPU NVIDIA GeForce GTX 1060 за 1,9 мс, в то время как на процессоре Intel Core i7-6700 с частотой 3,4 ГГц – за 151 мс.

Ключевые слова: изображения, локальные гистограммы, алгоритм, параллельная версия, CUDA

Для цитирования. Троцкий, Ф. С. Алгоритм быстрого вычисления локальных гистограмм изображения на видеокарте / Ф. С. Троцкий, Б. А. Залесский // Информатика. – 2019. – Т. 16, № 1. – С. 49–57.

Algorithm of fast computation of local image histograms on video card

Philip S. Trotski, Boris A. Zalesky✉

The United Institute of Informatics Problems
of the National Academy of Sciences of Belarus, Minsk, Belarus
✉*E-mail: zalesky@newman.bas-net.by*

Abstract. An algorithm of parallel computation of image histograms of different types, including brightness and oriented gradient ones, on video cards of various types is presented. Now local histograms are used for solution of some tasks of image processing and recognition, but their application is restricted due to the long computational time. One of the difficulties appearing during parallel computations of this vector feature is the large number of conflicts of simultaneous access to video memory sells. In the developed version, the number of conflicts of simultaneous access are many times reduced. It accelerated the computations. For instance, 9D vectors of histograms of oriented gradient for all 256×256 windows of a HD image are calculated on the

GPU NVIDIA GeForce GTX 1060 within 1,9 msec, whereas the same computations made by the CPU Intel Core i7-6700 take 151 msec.

Keywords: images, local histograms, algorithm, parallel version, CUDA

For citation. Trotski Ph. S., Zalesky B. A. Algorithm of fast computation of local image histograms on video card. *Informatics*, 2019, vol. 16, no. 1, pp. 49–57 (in Russian).

Введение. В последние годы гистограммы разной модальности [1, 2] используются для решения задач обработки и распознавания изображений, в том числе улучшения качества изображений, поиска в базах изображений, обнаружения пешеходов и автомобилей, лиц людей [2–7] и т. д., однако их вычисление требует выполнения большого числа операций и поэтому занимает достаточно продолжительное время. Определение гистограммы яркостей дается, например, в работе [1, с. 149].

В настоящей статье предлагается быстрая версия алгоритма вычисления локальных (оконных) гистограмм различных типов, предназначенная для выполнения на видеокартах, которые имеют программно-аппаратную архитектуру CUDA, и позволяющая вычислять гистограммы кадров видеопоследовательностей форматов HD и FullHD в режиме реального времени.

Обычно гистограммы яркости и цвета используются в качестве вспомогательных средств в задачах обработки изображений и в гораздо меньшей мере – в задачах распознавания в отличие от гистограмм ориентированного градиента (традиционное сокращение – HoG), которые применяются в основном при обнаружении, идентификации и распознавании [2].

В работе приводится описание алгоритма для случая вычисления гистограмм ориентированного градиента HoG. Вычисление оконных HoG отличается от вычисления гистограмм яркости необходимостью многократной проверки в каждом пикселе наличия ненулевого градиента, только в случае его присутствия следует приступать к изменению гистограммы. Многократная дополнительная проверка приводит к увеличению времени вычисления, поэтому вычисление гистограмм ориентированного градиента требует больше времени.

Непосредственное вычисление оконных гистограмм для всех пикселов изображения занимает недопустимо много времени, что делает невозможным их использование при решении ряда задач. Для повышения производительности параллельных вычислений набора оконных гистограмм были разработаны специальные приемы, уменьшившие на несколько порядков число конфликтов одновременного доступа к ячейкам памяти видеокарты, что позволило существенно ускорить расчеты.

Время вычисления девятимерных HoG изображения формата HD на процессоре Intel Core i7-6700 с частотой 3,4 ГГц занимает 151 мс, в то время как предложенный алгоритм, выполненный на GPU NVIDIA GeForce GTX 1060, справился с этой задачей за 1,9 мс. Малое время вычисления локальных гистограмм для всех пикселов изображения делает возможным применение быстрой версии алгоритма для работы с видеопоследовательностью. Для сравнения: параллельная версия алгоритма, реализованная в пакете OpenCV 3.3, вычисляет только набор девятимерных гистограмм решетки 8×8 пикселов этого же изображения и формирует набор дескрипторов, каждый из которых состоит из четырех гистограмм, в течение 1,33 мс (в данном случае отсутствуют конфликты общего доступа к памяти).

Обозначения. Известно несколько разных версий HoG полутонового изображения. Для определения одной из них будем использовать конечную решетку $\mathcal{P}=\{(x,y)\}$, $x=0,\ ...,\ m-1$, $y=0,\ ...,\ n-1$, пикселов $\mathbf{p}=(x,y)$; прямоугольные окрестности $O_{r,w}(\mathbf{p})$ размером $r\times w$, индексированные своими левыми верхними пикселами $\mathbf{p}$; полутоновые изображения $\mathbf{I}=\{I(\mathbf{p})\}$, $\mathbf{p}\in\mathcal{P}$, и градиент $\mathbf{G}_I(\mathbf{p})$ полутонового изображения $\mathbf{I}$, который представляет собой набор 2D-векторов $g(\mathbf{p})=\left(\frac{\partial}{\partial x}I(\mathbf{p}),\frac{\partial}{\partial y}I(\mathbf{p})\right)$, вычисленных каким-либо из известных способов и имеющих угол $\alpha(\mathbf{p})$ $(0\le\alpha(\mathbf{p})<360)$ с положительным направлением оси OX и длину $|g(\mathbf{p})|$. Потребуется также $L+1$-мерный вектор $\mathbf{b}=(b_0,\ldots,b_L)$, $b_0=0, b_L=360$, с помощью которого, соб-

ственно, определяются ячейки гистограммы оконного ориентированного градиента $\mathbf{H}(\mathbf{p}) = (h_0, \ldots, h_{L-1})$ с координатами

$$h_i(\mathbf{p}) = \sum_{\mathbf{q} \in O_{r,w}(\mathbf{p})} \mathbf{1}_{\{b_i \le \alpha(\mathbf{q}) < b_{i+1}\}}, \; 0 \le i < L, \tag{1}$$

для индикаторной функции $\mathbf{1}_{\{C\}}$, равной 1, если условие C выполняется, и равной нулю в противном случае. В некоторых случаях HoG определяется по формулам

$$h_i(\mathbf{p}) = \sum_{\mathbf{q} \in O_{r,w}(\mathbf{p})} |g(\mathbf{q})| \mathbf{1}_{\{b_i \le \alpha(\mathbf{q}) < b_{i+1}\}}, \quad h_i(\mathbf{p}) = \sum_{\mathbf{q} \in O_{r,w}(\mathbf{p})} \sqrt{|g(\mathbf{q})|} \mathbf{1}_{\{b_i \le \alpha(\mathbf{q}) < b_{i+1}\}}. \tag{2}$$

Для определенности будем рассматривать градиент $\mathbf{H}(\mathbf{p})$, найденный по формуле (1), полагая, что $L \le 360$ и $b_i = 360i/L, 0 \le i \le L$, хотя версии HoG, заданные формулой (2), могут быть вычислены аналогично.

Описание алгоритма. Вычисление на CPU компьютера локальных гистограмм, которые представляют собой векторы достаточно большой размерности, имеющие до 256 координат в случае гистограмм яркости или цвета и до 360 координат в случае гистограмм ориентированного градиента, не позволяет получить приемлемую скорость расчетов даже при параллельном использовании ядер процессоров. Специфика вычислений такова, что числовая характеристика каждого пиксела (например, яркость, цвет или угол наклона и длина градиента) используется для построения большого числа локальных гистограмм. Кроме того, к ячейкам гистограммы приходится обращаться многократно. В связи с этим возникает огромное число конфликтов одновременного доступа к ячейкам памяти, вызывающих длительные задержки вычислений, большая часть которых устраняется при выполнении предложенного алгоритма на GPU.

Еще одна особенность, которую пришлось учитывать при разработке алгоритма, – большая размерность выходных данных. Например, размерность массива 360-мерных HoG полутонового изображения размера Full HD равна примерно 746,5 Мб. Из пяти видов видеопамяти, доступных в современных видеокартах, только глобальная и текстурная имеют объем, достаточный для хранения вычисленных гистограмм, но при этом они обладают малой скоростью чтения (записи) данных. Наиболее быстрая разделяемая память доступна только внутри каждого блока видеокарты, и ее объем не превосходит 64 Кб [8]. Для ускорения вычислений пришлось оптимизировать стратегию использования глобальной и разделяемой памяти.

Были построены, реализованы и протестированы несколько версий параллельного вычисления HoG. Ниже приведена версия, обеспечившая наибольшее быстродействие для окон $O_{r,w}(\mathbf{p})$ «не очень большого» размера. Кавычки использованы в предыдущем предложении потому, что для разных видеокарт оптимальные размеры окон могут отличаться. Так, например, для видеокарт NVIDIA GeForce GTX 1050, 1060 наибольшее быстродействие предлагаемой версии вычисления HoG достигается на окнах размером $r, w \le 16 \times 16$. Сначала опишем эту версию, а потом, опираясь на нее, версию, обеспечивающую большее быстродействие на больших окнах [9].

Для уменьшения числа конфликтов одновременного доступа к памяти и объема вычислений сначала рассчитывались и сохранялись в глобальной памяти промежуточные HoG $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ окон $\mathbf{U}(\mathbf{p})$ размером $1 \times w$, представляющих собой горизонтальные растровые отрезки пикселов:

$$\mathbf{U}(\mathbf{p}) = \{\mathbf{p} + (0, i) \mid 0 \le i < w\}.$$

Затем искомые $\mathbf{H}(\mathbf{p})$ вычислялись по формуле

$$\mathbf{H}(\mathbf{p}) = \sum_{j=0}^{r-1} \mathbf{H}_{\mathrm{U}}(\mathbf{p} + (j, 0)).$$

Такой способ вычисления позволяет сократить объем операций примерно в r раз, так как каждая промежуточная HoG $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ участвует в вычислении r искомых HoG $\mathbf{H}(\mathbf{p})$.

Непосредственное параллельное вычисление всех вспомогательных HoG $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ также приводит к возникновению большого количества конфликтов одновременного доступа к памяти видеокарты, поэтому был использован следующий способ распараллеливания: сначала в параллельном режиме рассчитывались вспомогательные HoG $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ только для $\mathbf{p}$, лежащих на вертикальных направляющих, т. е. для пикселов вида $\mathbf{p}+(\mu, vw)$, где $0 \le \mu < m, 0 \le \nu < \frac{n}{w}$, а затем для остальных пикселов методом бегущей строки [10].

Для расчета промежуточных гистограмм, индексированных пикселами, лежащими на вертикальных направляющих, выделялось по одной нити (одному потоку) и одному массиву в разделяемой памяти блока на каждую гистограмму $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$.

После завершения вычисления промежуточной гистограммы $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ окна $O_{1,w}(\mathbf{p})$ с первым пикселом $\mathbf{p}$ на направляющей выделенная для этого нить $t(\mathbf{p})$ рассчитывала последовательно промежуточные гистограммы $\mathbf{H}_{\mathrm{U}}(\mathbf{p}+(0,i)), 1 \le i < w,$ методом бегущей строки.

Обозначим для краткости пиксел $\mathbf{p}_i = \mathbf{p}+(0,i)$. Для вычисления $\mathbf{H}_{\mathrm{U}}(\mathbf{p}_i), 1 \le i < w$, той же нитью $t(\mathbf{p})$ проверялось условие наличия градиента в пикселе $\mathbf{p}_{i-1}$, т. е. условие $g(\mathbf{p}_{i-1}) \ne 0$. В случае его выполнения нить $t(\mathbf{p})$ вычитала из координаты ℓ гистограммы $\mathbf{H}_{\mathrm{U}}(\mathbf{p}_{i-1})$, удовлетворяющей неравенствам $b_\ell \le \alpha(\mathbf{p}_{i-1}) < b_{\ell+1}$, единицу. Затем та же нить проверяла условие $g(\mathbf{p}_{i-1+w}) \ne 0$ и в случае его выполнения добавляла к координате ℓ гистограммы $\mathbf{H}_{\mathrm{U}}(\mathbf{p}_{i-1})$, удовлетворяющей условию $b_\ell \le \alpha(\mathbf{p}_{i-1+w}) < b_{\ell+1}$, единицу. Преобразованная гистограмма, хранящаяся в выделенном массиве разделяемой памяти, записывалась в глобальную память как $\mathbf{H}_{\mathrm{U}}(\mathbf{p}_i)$. Видно, что каждая нить $t(\mathbf{p})$ вычисляла w промежуточных HoG методом бегущей строки, затрачивая на это $3w-2$ операции вместо w^2 при непосредственном вычислении.

Схема выделения нитей для вычисления $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ и вспомогательных гистограмм выделенными нитями изображена на рис. 1.

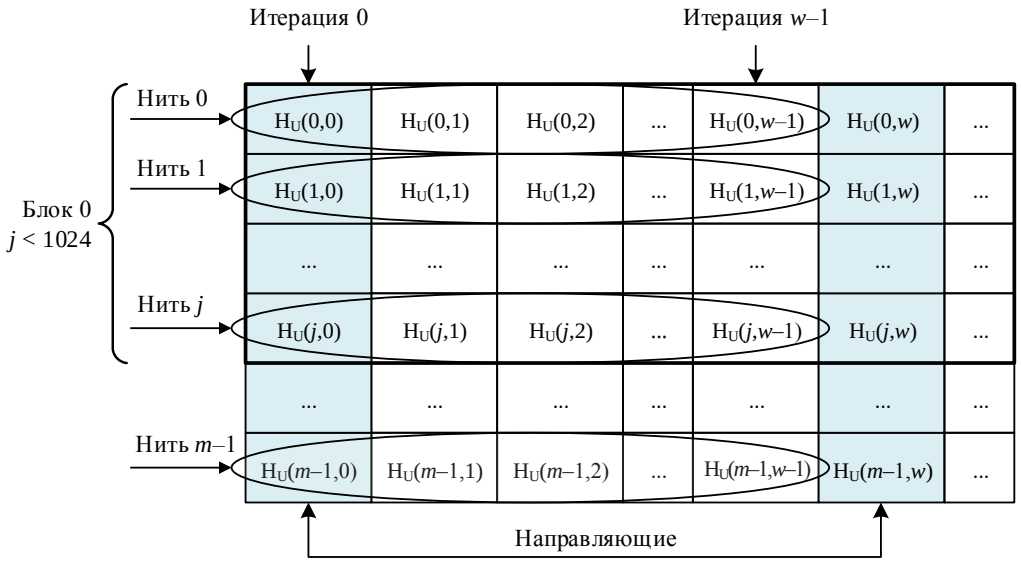

Рис. 1. Схема вычислений $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$

Для определения количества блоков N_{block}, нужных для вычисления вспомогательных гистограмм, обозначим через N_{shared} размер разделяемой памяти одного блока в байтах (обычно он не превышает 64 Кб), N_{thread} – количество нитей в блоке, b – размер в байтах переменных типа int, определяемых на видеокарте (обычно $b=4$), m_{bins} – число ячеек в гистограммах. Тогда количество промежуточных гистограмм $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ с $\mathbf{p}$, лежащими на направляющих, которые можно вычислить одним блоком, определяется выражением

$$L_{histinblock} = \min\left(\left[\frac{N_{shared}}{b \cdot m_{bins}}\right], N_{thread}\right). \tag{3}$$

Количество блоков, требуемых для вычисления всех вспомогательных гистограмм в случае, когда $m/L_{histinblock}$ не является целым числом, находится по формуле

$$N_{block} = \left(\left[\frac{m}{L_{histinblock}}\right]+1\right)\left[\frac{n}{w}\right], \tag{4}$$

где m – высота исходного изображения, n – его длина, а в случае, когда является целым числом, – по формуле

$$N_{block} = \frac{m}{L_{histinblock}}\left[\frac{n}{w}\right]. \tag{5}$$

Гистограммы $\mathbf{H}(\mathbf{p})$ вычислялись с помощью аналогичного приема. В 2D-массиве глобальной памяти, хранящем промежуточные гистограммы $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$, выделялись горизонтальные направляющие на расстоянии r, равном высоте окна $O_{r,w}(\mathbf{p})$. Для вычисления каждой HoG $\mathbf{H}(\mathbf{p})$ с пикселом $\mathbf{p}$, лежащим на горизонтальной направляющей, выделялась одна нить $t(\mathbf{p})$, которая суммировала r вспомогательных гистограмм (рис. 2):

$$\mathbf{H}(\mathbf{p}) = \mathbf{H}_{\mathrm{U}}(\mathbf{p}) + \ldots + \mathbf{H}_{\mathrm{U}}(\mathbf{p}+(r-1,0)).$$

Далее эта же нить вычисляла гистограммы $\mathbf{H}(\mathbf{p}+(1,0)),\ldots,\mathbf{H}(\mathbf{p}+(r-1,0))$ методом бегущей строки.

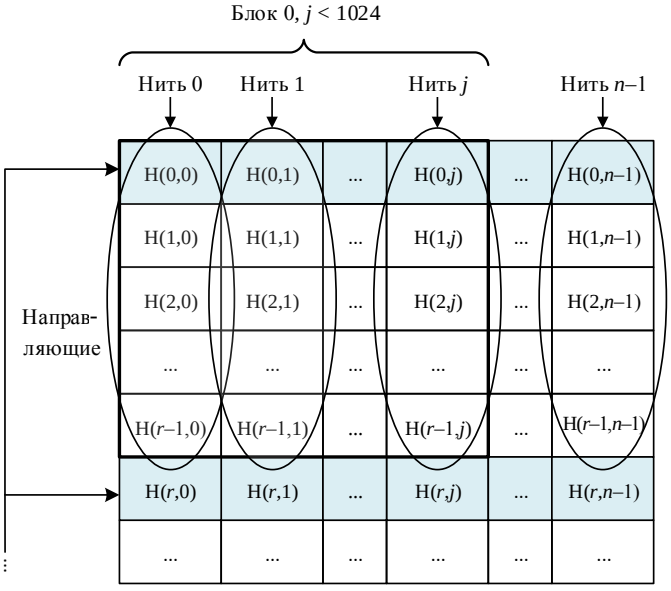

Рис. 2. Схема вычислений $\mathbf{H}(\mathbf{p})$

Для некоторых сочетаний размеров окон и количества ячеек в гистограмме лучший результат был получен в случае, когда вспомогательные $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ и искомые $\mathbf{H}(\mathbf{p})$ вычислялись не одной, а несколькими нитями. Для этого осуществлялись следующие действия:

– каждой нити выделялся отдельный массив быстрой разделяемой памяти;

– окна $O_{1,w}(\mathbf{p})$ и $O_{r,w}(\mathbf{p})$ делились на несколько частей;

– каждая нить вычисляла $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ или $\mathbf{H}(\mathbf{p})$ своей части окна в выделенном для нее массиве;

– несколько первых нитей из выделенных для вычисления конкретной HoG рассчитывали $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ или $\mathbf{H}(\mathbf{p})$ для всего окна методом редукции [11].

Для подсчета числа требуемых блоков обозначим k_{reduce} количество нитей, выделяемых для нахождения каждой $\mathbf{H}_{\mathrm{U}}(\mathbf{p})$ или $\mathbf{H}(\mathbf{p})$, и воспользуемся формулами (3)–(5), в которые вместо N_{shared} и N_{thread} подставим N_{shared} / k_{reduce} и N_{thread} / k_{reduce}.

Анализ быстродействия разработанного алгоритма. При оценке быстродействия предложенной параллельной версии алгоритма вычисления гистограмм были проведены эксперименты по нахождению HoG, занимающему больше времени по сравнению с вычислением гистограмм яркости, для полутоновых изображений разных размеров на персональном компьютере, оборудованном видеокартой NVIDIA GeForce GTX 1050 с 2 Гб видеопамяти, и лэптопе с видеокартой NVIDIA GeForce GTX 1060 с 6 Гб видеопамяти. Для тестирования использовались изображения природного ландшафта, а также искусственно созданные изображения, заполненные градиентом (рис. 3).

Рис. 3. Примеры тестовых изображений

Оценка времени вычисления HoG проводилась следующим образом: гистограммы ориентированного градиента каждого тестового изображения вычислялись последовательно 300 раз, затем вычислялось среднее из имеющихся 300 значений, после этого полученные значения осреднялись по количеству тестовых изображений и результат заносился в таблицу. Время в таблице дано в миллисекундах, в числителях дробей приведены результаты работы первой видеокарты, а в знаменателях – второй.

Результаты экспериментов. Размер полутонового изображения 1280×720, NVIDIA GeForce GTX 1050/1060

Размер окна HoG	Количество ячеек HoG							
	9		30		60		90	
	Время, мс	Редукция	Время, мс	Редукция	Время, мс	Редукция	Время, мс	Редукция
8×8	2,6/1,5	4/4	12,3/6,9	1/2	24,8/14,0	1/1	45,2/25,1	1
16×16	3,0/1,7	4/4	14,2/7,5	1/4	26,3/14,7	1/2	46,1/27,5	1
32×32	3,2/1,8	4/8	15,2/7,7	2/4	27,0/14,9	1/2	46,5/28,0	1
128×128	3,3/1,8	8/8	14,3/7,1	4/4	25,0/13,7	2/2	42,3/26,6	1
256x256	3,2/1,9	8/8	11,9/5,9	4/4	19,6/10,8	2/2	34,8/21,3	1

Время выделения памяти на видеокарте не включено в результаты вычислений, так как при повторном использовании алгоритма целесообразно применять однажды выделенную память. Время копирования вычисленных девятимерных HoG изображения размером 1280×720 из видеопамяти GPU NVIDIA GeForce GTX 1050 в память компьютера составляет 2,8 мс.

На рис. 4 показаны графики зависимости времени вычисления HoG с 9, 30, 60 и 90 ячейками от размеров окна. Время вычисления уменьшается потому, что HoG не вычисляется на краях изображения шириной в половину окна.

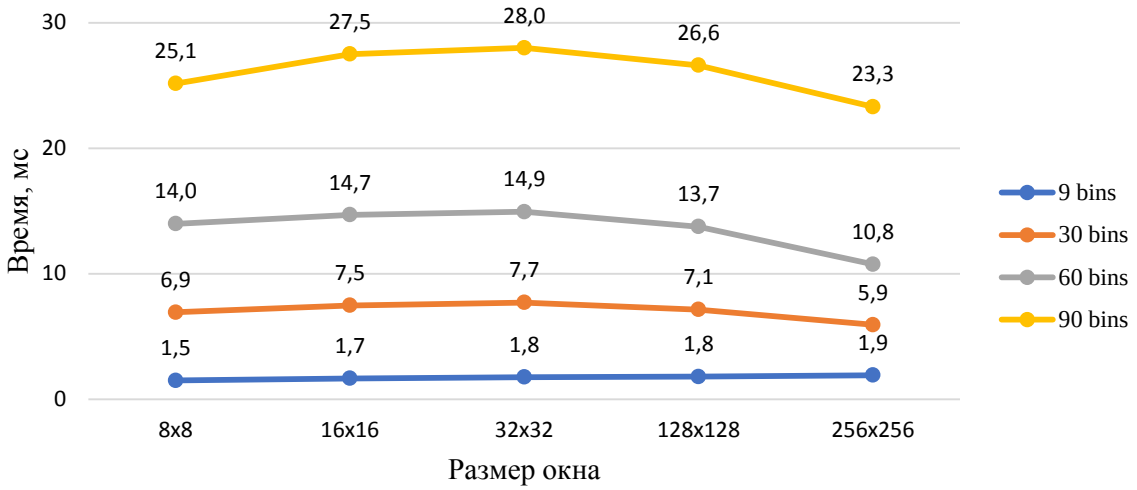

Рис. 4. Зависимость времени вычисления HoG предложенным алгоритмом от размера окна и числа ячеек (bins)

Полученные оценки быстродействия алгоритма сравнивались с быстродействием CUDA-версии функции вычисления HoG из открытого пакета OpenCV 3.3. Специфика указанной функции заключается в том, что HoG в ней вычисляется не для всех пикселов – изображение разбивается на непересекающиеся квадраты размером 8×8 и гистограммы ориентированного градиента вычисляются только для этих непересекающихся квадратов. Затем формируются векторы-дескрипторы квадратов изображения 16×16, вершины которых находятся на решетке с шагом 8×8, путем конкатенации четырех HoG квадратов 8×8, лежащих в выбранном окне 16×16, в один вектор. Таким образом, в окне 64×128 OpenCV 3.3 вычисляет 105 дескрипторов, каждый из которых состоит из четырех HoG. Отметим следующее: во-первых, количество гистограмм, вычисляемых реализацией предложенного алгоритма, в 64 раза больше количества гистограмм, вычисляемых упомянутой функцией, и, во-вторых, функция HOGDescriptor::compute в отличие от предложенного алгоритма вычисляет HoG непересекающихся окон. При таких вычислениях практически не возникает конфликтов одновременного доступа к памяти.

Известно, что HoG применяется при решении задач распознавания изображений. Построенный алгоритм тестировался при решении задачи распознавания лиц. На рис. 5 приведены результаты решения задачи распознавания лиц из тестового набора AT&T с помощью HoG, вычисленных с использованием предложенного алгоритма, и алгоритмов Eigenface и LBPH, реализованных в OpenCV 3.3.

Рис. 5. Примеры изображений лица одного человека из тестового набора AT&T

Более точно с помощью HoG решалась задача идентификации лица по одному тестовому изображению на основе поиска в базе данных, содержащей одно изображение этого лица, причем отличное от тестового. В тестовом наборе AT&T содержится 400 изображений 40 лиц размером 92×112 пикселов, при этом имеется 10 изображений каждого лица. Было проведено 3600 экспериментов – по 10 экспериментов с каждым лицом. При использовании HoG было идентифицировано правильно 68,64 % лиц, в то время как программная реализация алгоритма Eigenface и LBPH из OpenCV 3.3 на тех же наборах данных нашла 55,89 и 56,97 % правильных решений.

Заключение. Предложенный алгоритм быстрого вычисления оконных гистограмм разрабатывался с целью получения гистограмм для всех возможных положений окна на изображении за как можно меньшее время. Непосредственное вычисление гистограмм на процессоре или даже на видеокарте в силу специфики задачи занимает очень большое время, что приводит в большинстве случаев к невозможности применения гистограммных методов в быстрых приложениях, например таких, как обработка и распознавание объектов на видеопоследовательностях. Разработанный способ вычисления оконных гистограмм, предназначенный для реализации на видеокартах, позволил значительно ускорить их получение, в том числе и при работе с видеопоследовательностями в режиме реального времени.

Список использованных источников

1. Gonzalez, R. Цифровая обработка изображений / R. Gonsalez, R. Woods. – M. : Техносфера, 2005. – 1070 с.

2. Dalal, N. Histograms of oriented gradients for human detection / N. Dalal, B. Triggs // Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'2005). – San Diego, 2005. – Vol. 1. – P. 886–893.

3. Rotation invariant histogram of oriented gradients / M. Cheon [et al.] // Intern. J. of Fuzzy Logic and Intelligent Systems. – 2011. – Vol. 11, no. 4. – P. 293–298.

4. Marimon, D. Orientation histogram-based matching for region tracking / D. Marimon, T. Ebrahimi // Eighth Intern. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07). – Santorini, Greece, 2007. – P. 8.

5. Ragb, H. Histogram of oriented phase and gradient (HOPG) descriptor for improved pedestrian detection / H. Ragb, V. Asari // IS&T Intern. Conf. on Electronic Imaging: Video Surveillance and Transportation Imaging Applications. – San Francisco, 2016.

6. Ragb, H. Histogram of oriented phase (HOP): a new descriptor based on phase congruency / H. Ragb, V. Asari // Proc. SPIE 9869, Mobile Multimedia/Image Processing, Security, and Applications. – Bellingham, 2016. – Vol. 98690.

7. Facial expression recognition based on facial components detection and HOG features / J. Chen [et al.] // Scientific Cooperations Intern. Workshops on Electrical and Computer Engineering Subfields, 22–23 Aug. 2014, Istanbul, Turkey. – Istanbul, 2014. – P. 64–69.

8. Harris, M. Using Shared Memory in CUDA C/C++ [Electronic recource] / M. Harris. – Mode of access: https://devblogs.nvidia.com/using-shared-memory-cuda-cc. – Date of access: 26.09.2018.

9. Залесский, Б. А. Параллельная версия детектора экстремальных особых точек / Б. А. Залесский, Ф. С. Троцкий // Информатика. – 2018. – Т. 15, № 2. – С. 55–63.

10. Булашев, С. В. Статистика для трейдеров / С. В. Булашев. – М. : Компания Спутник+, 2003. – 245 с.

11. Сандерс, Д. Технология CUDA в примерах: введение в программирование графических процессоров : пер. с англ. А. А. Слинкина / Д. Сандерс, Э. Кэндрот. – М. : ДМК Пресс, 2013. – 234 с.

References

1. Gonzalez R., Woods R. Cifrovaya obrabotka izobrazhenij [*Digital Image Processing*]. Moscow, Tehnosfera, 2005, 1070 p.

2. Dalal N., Triggs B. Histograms of oriented gradients for human detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*. San Diego, 2005, vol. 1, pp. 886–893.

3. Cheon M., Lee W., Hyun C.-H., Park M. Rotation invariant histogram of oriented gradients. *International Journal of Fuzzy Logic and Intelligent Systems*, 2011, vol. 11, no. 4, pp. 293–298.

4. Marimon D., Ebrahimi T. Orientation histogram-based matching for region tracking. *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07)*. Santorini, Greece, 2007, p. 8.

5. Ragb H., Asari V. Histogram of oriented phase and gradient (HOPG) descriptor for improved pedestrian detection. *IS&T International Conference on Electronic Imaging: Video Surveillance and Transportation Imaging Applications*. San Francisco, 2016.

6. Ragb H., Asari V. Histogram of oriented phase (HOP): a new descriptor based on phase congruency. *Proceedings SPIE 9869, Mobile Multimedia/Image Processing, Security, and Applications*. Bellingham, 2016, vol. 98690. DOI: 10.1117/12.2225159

7. Chen J., Chen Z., Chi Z., Fu H. Facial expression recognition based on facial components detection and HOG features. *Scientific Cooperations International Workshops on Electrical and Computer Engineering Subfields, 22–23 August 2014, Istanbul, Turkey*. Istanbul, 2014, pp. 64–69.

8. Harris M. *Using Shared Memory in CUDA C/C++*. Available at: https://devblogs.nvidia.com/using-shared-memory-cuda-cc (accessed 26.09.2018).

9. Zalesky B. A., Trotski Ph. S. Parallelnaya versiya detektora ekstremalnyih osobyih tochek [Parallel version of detector of extremal key points]. Informatika [*Informatics*], 2018, vol. 15, no. 2, pp. 55–63 (in Russian).

10. Bulashev S. V. Statistika dlya treyderov. *Statistics for traders*. Moscow, Kompanija Sputnik+, 2003, 245 p. (in Russian).

11. Sanders D., Kendrot E. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Boston, Addison-Wesley Professional, 2010, 320 p.

Информация об авторах

Троцкий Филипп Сергеевич, младший научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск, Беларусь.
E-mail: trotskiphilipp@gmail.com

Залесский Борис Андреевич, доктор физико-математических наук, заведующий лабораторией обработки и распознавания изображений, Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск, Беларусь.
E-mail: zalesky@newman.bas-net.by

Information about the authors

Philip S. Trotski, Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, Belarus.
E-mail: trotskiphilipp@gmail.com

Boris A. Zalesky, Dr. Sc. (Phys.-Math.), Head of the Laboratory of Image Processing and Recognition, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, Belarus.
E-mail: zalesky@newman.bas-net.by