ПРОГРАММИРОВАНИЕ, 2021, № 5, с. 3–11

АНАЛИЗ И ТРАНСФОРМАЦИЯ ПРОГРАММ

УДК 004.932.2

ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ХАФА С ИСПОЛЬЗОВАНИЕМ СОПРОЦЕССОРА СРСА

© 2021 г. Ф. А. Аникеев[a,*], Г. О. Райко[a,], Е. Е. Лимонова[b,c,***], М. А. Алиев[b,c,****], Д. П. Николаев[b,d,*****]**

[a] *ФГУ "Федеральный научный центр Научно-исследовательский институт системных исследований" РАН, 117218 Москва, Нахимовский проспект, д. 36, к. 1, Россия*

[b] *ООО "Смарт Энджинс Сервис", 117312 Москва, проспект 60-летия Октября, д. 9, Россия*

[c] *Федеральный исследовательский центр "Информатика и управление" РАН, 119333 Москва, ул. Вавилова. д. 44, кор. 2, Россия*

[d] *Институт проблем передачи информации им. А.А. Харкевича РАН, 127051 Москва, Большой Каретный переулок, д. 19, стр. 1, Россия*

**E-mail: snowfed@gmail.com*

***E-mail: raiko@niisi.msk.ru*

****E-mail: limonova@smartengines.ru*

*****E-mail: aliev@smartengines.ru*

******E-mail: dimonstr@iitp.ru*

Поступила в редакцию 02.02.2021 г.
После доработки 19.03.2021 г.
Принята к публикации 25.04.2021 г.

В работе строится вычислительно эффективная реализация алгоритма Брейди расчета быстрого преобразования Хафа (БПХ) на отечественном сопроцессоре СРСА, входящем в состав системы-на-кристалле 1890ВМ9Я "КОМДИВ128-М". Показывается, что БПХ находит широкое применение в задачах анализа изображений, от зрительных систем беспилотного транспорта до вычислительной рентгеновской томографии. Приводится и анализируется с точки зрения низкоуровневой имплементации классическая рекурсивная реализация БПХ. Впервые рассматривается более эффективный нерекурсивный вариант алгоритма, для которого проводится анализ нагрузки на вычислители и память сопроцессора, а также экспериментальные замеры производительности. Показывается, что теоретически возможная производительность нерекурсивного алгоритма на СРСА составляет 800 Мопс, при этом максимально достижимая на практике производительность составила 470 Мопс, а максимальное полученное экспериментально значение оказалось 406 Мопс. При этом загрузка вычислителей сопроцессора достигла 18%. Таким образом, несмотря на относительно малое число арифметических операций в методе, использование сопроцессора оказывается целесообразным.

DOI: 10.31857/S0132347421050022

1. ВВЕДЕНИЕ

Задачи искусственного интеллекта в области технических зрительных систем отличаются большими объемами однородной информации, которые необходимо обработать для принятия даже самых элементарных решений. При этом зрительные системы эксплуатируются, как правило, в режиме реального времени, что не характерно для большинства иных областей, работающих с "большими данными". Требования к быстрой обработке больших объемов данных приводят к тому, что классические высокопроизводительные алгоритмы (наряду с популярными нейросетевыми методами) продолжают занимать важное место в анализе зрительной информации. В данной работе мы сосредоточимся на одном весьма известном вычислительно эффективном инструменте анализа изображений, а именно – алгоритме Брейди вычисления быстрого приближенного дискретного преобразования Радона [1], часто называемом также быстрым преобразованием Хафа (БПХ) [2].

Исходно П. Хаф позиционировал свой метод как способ идентификации прямолинейных треков в пузырьковой камере [3]. Поэтому наиболее известное приложение преобразования Хафа – это нахождение контрастных прямых или их отрезков на изображении [4]. Но преобразование

Хафа – это еще и дискретизация преобразования Радона, линейного разложения, имеющего фундаментальное математическое значение. Преобразование Радона имеет число приложений едва ли меньшее, чем преобразование Фурье. Помимо выявления графических примитивов БПХ используется для решения самых разных задач анализа изображений. В качестве примеров областей использования БПХ, весьма далеких от поиска отрезков, приведем бинаризацию [5] и сегментацию [6] изображений, а также задачу автоматического определения параметров оптических аберраций [7].

Как и в случае преобразования Фурье, при использовании преобразования Хафа важно наличие быстрых алгоритмов его вычисления. Использование БПХ тем более оправдано, чем выше требования к вычислительной эффективности. К зрительным системам беспилотного транспорта предъявляются особенно жесткие требования по быстродействию. Аппаратное оснащение таких систем может быть зафиксировано из соображений массы, габаритов и энерговооруженности, поэтому вычислительная оптимизация используемых алгоритмов на конкретном вычислителе становится крайне важна. В результате БПХ стало практически неотъемлемой частью таких систем. Чаще всего БПХ применяется для поиска элементов дорожной разметки [8–10], причем многие авторы создают алгоритмы, предназначенные для встраиваемых устройств [11–13]. Также БПХ может использоваться для прослеживания колес автомобилей [14] или в системах анализа транспортных потоков [15].

Проблема энергоэффективности является ключевой и для систем, работающих на мобильных устройствах [16], и в них также активно применяется БПХ. Системы распознавания документов используют его для поиска четырехугольника документа [17] или для поиска паттернов, состоящих из параллельных отрезков. Алгоритмы поиска параллельных пучков, использующие БПХ, позволяют найти наклон документа на изображении [18], текстовой строки [19] или отдельных символов в ней [20]. Еще одной задачей распознавания на мобильных устройствах, в которой применяется БПХ, является обнаружение штрихкодов [21].

По несколько иным причинам БПХ используется в вычислительной рентгеновской томографии [22, 23]. Здесь при значительных привлекаемых вычислительных ресурсах до сих пор не достигнуты желаемые параметры разрешения реконструкции, то есть объемы обрабатываемых данных постоянно растут, что при использовании асимптотически неэффективных алгоритмов рано или поздно приводит к нехватке ресурсов. Напротив, использование правильной алгоритмической базы позволяет гораздо более гибко подходить к вопросам используемого оборудования. Так, в [24] была обоснована возможность создания компьютерного томографа на отечественной элементной базе при использовании алгоритмов реконструкции, использующих БПХ. Кроме того, задача ускорения реконструкции крайне актуальна для нового класса онлайн-методов, позволяющих в режиме реального времени оценить и скорректировать параметры измерений [25].

Следует также отметить, что использование классических алгоритмов с БПХ не противоречит развитию нейросетевых методов, а позволяет сделать их более эффективными с вычислительной точки зрения. На сегодняшний день новым направлением в распознавании является построение так называемых Хаф-сетей на основе классических сверточных. В состав таких сетей включается необучаемый слой, вычисляющий дискретное преобразование Радона алгоритмом БПХ. Этот слой позволяет извлекать информацию о прямых и отрезках на изображении даже при условии небольших рецептивных слоев сверточных нейронов, в отличие от классических сверточных слоев, которые способны при таком количестве слоев и связей анализировать лишь локальные признаки [26–29]. Проще всего этот эффект демонстрируется при сравнении нейросетевых методов поиска отрезков [30]. Но и в серьезных практических задачах [27, 31, 32] использование Хаф-архитектур также позволяет улучшить соотношение точности распознавания и требуемого объема вычислений.

Таким образом, задача эффективной реализации БПХ на различных вычислительных платформах представляет огромный интерес. В данной работе исследована возможность реализации алгоритма для вычисления БПХ по методу Брейди на микропроцессоре 1890ВМ9Я, в состав которого входит быстрый сопроцессор комплексной арифметики, СРСА.

2. РЕКУРСИВНЫЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ БПХ ПО МЕТОДУ БРЕЙДИ

Пусть дано изображение размером $N \times N$, причем $N = 2^k$, где k – целое. Алгоритм Брейди [1] позволяет за $4N^2 \log_2 N$ операций суммирования построить массив размера $4N \times N$, каждый элемент которого содержит сумму значений входного массива по дискретному паттерну, приближающему прямую с параметрами, задаваемыми координатами элемента выходного массива.

Используя нормальную параметризацию, обозначим координаты прямой l, вдоль которой требуется вычислить сумму, как (φ_l, ρ_l). При вычис-

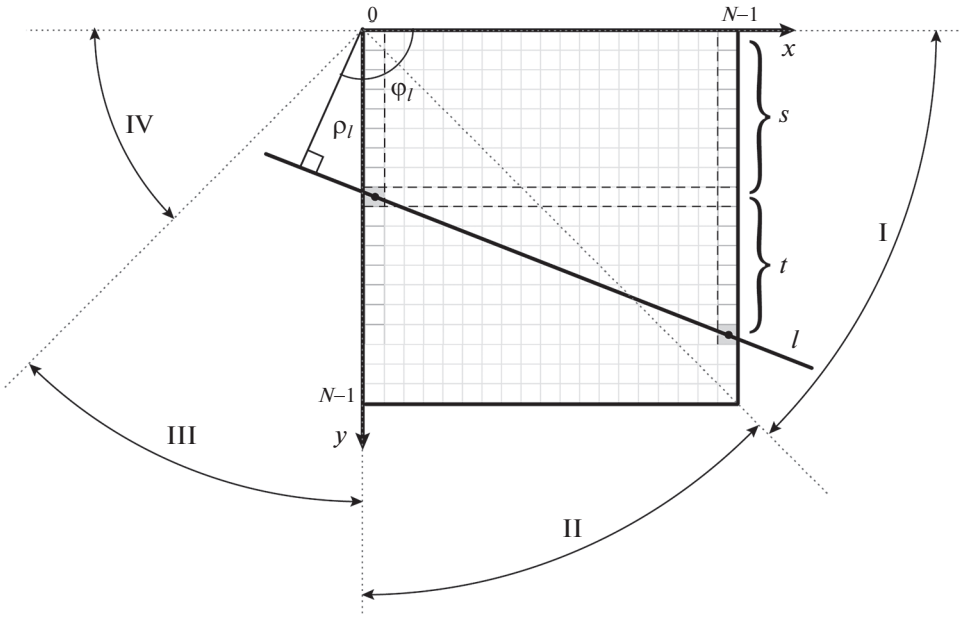

Рис. 1. Связь между нормальными параметрами (φ_l, ρ_l) прямой и координатами (s, t). Римскими цифрами обозначены 4 угловых диапазона. Прямая l относится к диапазону III.

лении БПХ методом Брейди суммы вдоль прямых с φ_l в четырех диапазонах ([0°, 45°], [45°, 90°], [90°, 135°] и [135°, 180°]) рассчитываются независимо, формируя подмассивы размера $N \times N$.

Рассмотрим третий угловой диапазон, $\varphi_l \in [90°, 135°]$. В ячейке с координатами (s, t) соответствующего подмассива алгоритмом Брейди формируется сумма вдоль прямой, проходящей через центр ячейки s нулевого столбца и центр ячейки $s + t$ последнего столбца входного массива (см. рис. 1). Нормальные координаты соответствующей прямой связаны с координатами ячейки следующими соотношениями:

$$\begin{cases} \varphi_l = \operatorname{arctg}\dfrac{t}{N-1} + \dfrac{\pi}{2} \\ \rho_l = s\dfrac{N-1}{\sqrt{t^2 + (N-1)^2}} \end{cases}. \qquad (2.1)$$

Для других диапазонов углов система координат (s, t) вводится симметричным образом. В такой параметризации на изображении размера $N \times N$ существуют N^2 дискретных прямых в каждом угловом диапазоне. Поэтому прямая реализация дискретного преобразования Радона в ней имеет вычислительную сложность $O(N^3)$ [1]. В тех же условиях сложность метода Брейди составляет $O(N^2 \log N)$.

Следует заметить, что расположение входных ячеек, результат суммирования которых будет записан алгоритмом Брейди в ячейку (s, t), не наилучшим образом приближает идеальные прямые. Впрочем, известная оценка сверху на отклонение ячейки, участвующей в суммации, от идеальной прямой – $(\log_2 N)/6$ по одной из координат [33] – позволяет пренебречь этим отклонением в большинстве практических приложений. На рис. 2 для сравнения изображено максимальное (среди всех значений t) отклонение паттерна Брейди от идеала для линейного размера изображения $N = 8$.

В Листинге 1 приведена рекурсивная реализация алгоритма Брейди на языке MATLAB, опубликованная в работе [34]. В этой реализации на вход алгоритма, помимо изображения m, подается параметр *sign*, позволяющий выбрать из второго (*sign* = –1) и третьего (*sign* = +1) угловых диапазонов. Кроме того, приведенный код позволяет обрабатывать изображения любого размера, хотя случай размеров, отличных от степеней двойки, по-видимому, никогда содержательно не исследовался.

В рассматриваемой реализации алгоритм состоит из двух частей. Первая – рекурсивное разбиение изображения на части – левую и правую – до тех пор, пока мы не разобьем изображение на N векторов, где N – ширина изображения. Вторая – объединение пар таких частей в одно целое, до тех пор, пока не получится искомый Хаф-образ.

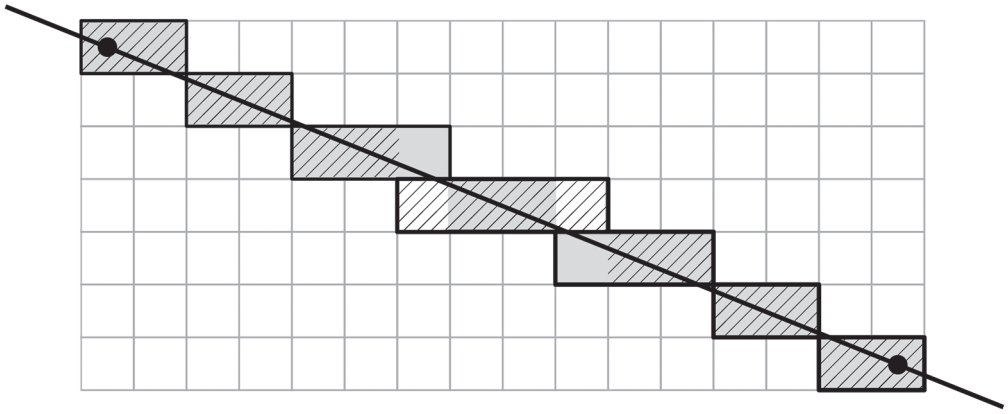

Рис. 2. Взаимное расположение идеальной прямой, ее наилучшей дискретизации и дискретизации Брейди для $N = 16$, $t = 6$. Серой заливкой обозначена наилучшая дискретизация, штриховкой – дискретизация Брейди.

Листинг 1: Рекурсивная реализация БПХ методом Брейди на языке MATLAB, опубликованная в [34].

```
 1 function h = fht2core(m, sign)
 2   n = size(m, 2);
 3   if n < 2
 4     h = m;
 5     return
 6   end
 7   n0 = floor(n/2);
 8   h = mergeHT(fht2core(m(:, 1 : n0, :), sign),
       ↪ fht2core(m(:, n0 + 1 : n, :), sign), sign);
 9 end
10
11 function h = mergeHT(h0, h1, sign)
12   [m, n0, o] = size(h0);
13   n1 = size(h1, 2);
14   n = n0 + n1;
15   h = zeros(m, n, o);
16   r0 = (n0 - 1) / (n - 1);
17   r1 = (n1 - 1) / (n - 1);
18   for i = 1 : n
19     t = i - 1;
20     t0 = round(t * r0);
21     t1 = round(t * r1);
22     s = mod(sign * (t - t1), m);
23     h(:, i, :) = h0(:, t0 + 1, :) + [h1(s + 1 : m, t1 + 1,
         ↪ :); h1(1 : s, t1 + 1, :)];
24   end
25 end
```

Это последнее объединение реализуется многократным выполнением одной и той же векторной операции над разными данными. Операция заключается в сложении двух векторов, один из которых циклически сдвинут (строка 23 Листинга 1). Таким образом, алгоритм БПХ, как и исходное преобразование Хафа, сводится к многократному выполнению векторной операции суммирования, но позволяет получить искомую аппроксимацию образа за меньшее число операций.

Алгоритм Брейди носит явно выраженный векторный характер, что позволяет надеяться на

эффективное использование массовых операций на сопроцессоре. Тем не менее, при реализации БПХ на языке программирования общего назначения требуется устранить некоторые особенности реализации алгоритма в MATLAB:

• Развернуть рекурсию в цикл для большего контроля за использованием памяти и минимизации накладных расходов.

• Исключить избыточное копирование данных.

3. НЕРЕКУРСИВНЫЙ АЛГОРИТМ

Несложно видеть, что рекурсивная процедура fht2core из Листинга 1 фактически строит полное двоичное дерево из интервалов по горизонтальному индексу изображения. Далее, с каждым вызовом процедуры mergeHT, проводится объединение двух интервалов и удаляются два листа с общим родителем, превращая последнего в лист. Так происходит поэтапное объединение всех интервалов, пока не останется один конечный интервал $[0, N-1]$, и мы не получим искомую аппроксимацию Хаф-образа.

Для изображения со стороной, не являющейся степенью двойки, затруднительно предугадать искомую структуру дерева. К счастью, построение такого дерева не составляет труда и состоит из вычисления горизонтальных индексов для интервалов, в соответствии с процедурой fht2core. Общее число интервалов – узлов полного двоичного дерева заранее известно и выражается через число листьев дерева. Число листьев есть число горизонтальных индексов изображения (его ширина N). В таком случае полное число интервалов составляет $2N-1$ [35].

Несмотря на то, что алгоритмически мы строим дерево, программно реализовать необходимо последовательное заполнение массива интервалов размером $2N-1$. После чего – двигаться по массиву в обратном направлении, объединяя интервалы в соответствии с процедурой mergeHT. Псевдокод заполнения массива интервалов I – структур с полями begin и end – приведен в Листинге 2, индексация начинается с 0.

Листинг 2: Процедура построения массива интервалов для использования в нерекурсивной реализации БПХ.

```
 1 I[0].begin := 0
 2 I[0].end := N
 3 jbeg := 0
 4 jend := 1
 5 while jbeg < jend
 6   j := jend
 7   while jbeg < jend
 8     n := I[jbeg].end - I[jbeg].begin
 9     if n > 2 do
10     n0 := floor(n / 2)
11     I[j].begin := I[jbeg].begin + n0
12     I[j].end := I[jbeg].end
13     j := j + 1
14     I[j].begin := I[jbeg].begin
15     I[j].end := I[jbeg].begin + n0
16     j := j + 1
17   jbeg := jbeg + 1
18 jend := j
```

Отметим, что в обратном проходе для каждого вызова аналога процедуры mergeHT будет извлекаться по два интервала с конца массива I. Таким образом, массив I здесь используется в качестве контейнера для реализации стека – основной структуры данных в рекурсивном варианте алгоритма.

4. ИСКЛЮЧЕНИЕ ИЗБЫТОЧНОГО КОПИРОВАНИЯ ДАННЫХ

В процедуре mergeHT два среза двумерного массива (два интервала горизонтальных индексов) объединяются в один. Размер результирующего среза h равен сумме размеров входных срезов $h0$ и $h1$.

Формально в процедуре mergeHT используется N столбцов дополнительной памяти, однако при реализации этой процедуры на языке программирования общего назначения количество дополнительной памяти можно существенно сократить. Для этого все манипуляции со столбцами следует преобразовать в манипуляции с указателями. При этом само изображение представляется в виде массива указателей, а для того, чтобы поменять местами два столбца, достаточно поменять местами указатели в массиве.

После того, как все суммы с некоторым столбцом из $h0$ или $h1$ были вычислены, этот столбец сам может быть использован для записи результатов. Тогда выполнить объединение двух срезов удается с использованием всего двух дополнительных столбцов.

5. СОПРОЦЕССОР СРСА

Сопроцессор комплексной арифметики СРСА в составе системы-на-кристалле (СнК) 1890ВМ9Я [36] – отмасштабированная на технологию 65 нм версия аналогичного блока СнК 1890ВМ7Я. СРСА имеет отдельную накристальную память команд (64 Кбайт), единый вычислительный конвейер, способный выбирать на исполнение одну вычислительную и одну команду пересылки и управления за такт и четыре вычислительных секции – 4 SIMD. Каждая вычислительная секция содержит 64 Кбайт накристальной памяти данных и набор из 64 64-разрядных регистров общего назначения. В составе сопроцессора имеется также набор разделяемых секциями 16 адресных регистров и 16 регистров общего назначения. К управляющим командам относятся также и команды чтения/записи накристальных памятей данных, скорость доступа к накристальным памятям – 16 байт/такт в каждой секции. СРСА выполнен как сопроцессор управляющего универсального процессора архитектуры MIPS64 Release 1. Загрузка и выгрузка данных из памятей секций осуществляется при помощи контроллера прямого доступа к памяти (ПДП), также заимствованного из СнК 1890ВМ7Я. Контроллер позволяет обмениваться данными с основной памятью со скоростью 16 байт/такт. Выполнение операций загрузки и выгрузки данных может быть совмещено с вычислениями. Последовательное изложение архитектуры СРСА приведено в [37].

Для программирования СРСА разработана библиотека цифровой обработки сигналов, которая содержит набор готовых к использованию оптимизированных для СРСА функций линейной алгебры и обработки сигналов, средства сбора статистики, примитивы для организации вычислений и пересылок, а также тесты – примеры использования указанных готовых функций и примитивов. Кроме этого, для обеспечения ускоренного процесса разработки, поддерживается кросс-вариант библиотеки, где СРСА эмулируется программно. Встроенный программный эмулятор обеспечивает потактовое выполнение программы внутри сопроцессора и эмуляцию работы контроллера ПДП, обслуживающего перемещение данных между системной памятью и памятями вычислительных секций [38].

Система команд СРСА содержит инструкции для работы с комплексными числами – парами 32-разрядных вещественных чисел формата ISO 60559. Каждая секция имеет 10 конвейеризированных сумматоров и умножителей, полностью задействованных в инструкции “бабочка Фурье”, которая использует помимо регистров еще и накристальную память коэффициентов. Инструкции “умножение вещественной 2 × 2-матрицы на вещественный 2d-вектор” и “комплексное умножение с накоплением”, использующие только регистры, задействуют до 8 сумматоров и умножителей. Таким образом, пиковая производительность СРСА при штатной тактовой частоте 800 МГц составляет 32 Гопс (10 оп./такт в 4 секциях при 800 МГц) или 25.6 Гопс при использовании регистровых инструкций.

Как было отмечено выше, ключевой операцией в алгоритме БПХ является поэлементное суммирование двух столбцов, где один операнд циклически сдвинут. На операции сложения двух векторов производительность СРСА не превышает 4.2 Гопс. Действительно, в то время как сумматоры способны выполнять по 2 сложения каждый такт в 4 секциях (команда сложения двух пар чисел – инструкция psadd), т.е. производительность сумматоров составляет 6.4 Гопс при 800 МГц, однако для достижения такого показателя производительности данная операция требует загрузки 4 чисел из локальной памяти и выгрузки 2 чисел в локальную память каждый такт в каждой секции, т.е. должна достигаться скорость обмена с локальной памятью не менее 19.2 Гчисел/сек при 800 МГц. Пропускная способность шины между сумматоров и накристальной памятью секции ограничена скоростью 4 числа/такт или 12.8 Гчисел/сек при 800 МГц. Таким образом, эффективная производительность сумматоров оказывается в 1.5 раза меньше пиковой.

Более того, контроллер ОЗУ типа DDR3 в составе СнК 1890ВМ9Я работает на частоте 400 МГц, что определяет пиковую скорость обмена между СРСА и ОЗУ в 6.4 Гбайт/сек (16 байт/такт при 400 МГц) или 1.6 Гчисел/сек. При таком ограничении производительность СнК в целом на операции сложения двух векторов не превысит 533 Мопс (3 числа/оп. при скорости 1.6 Гчисел/сек). Следует понимать, что это пиковая теоретическая производительность рассматриваемой операции, на практике такая производительность недостижима.

Таким образом, загрузка СРСА при выполнении быстрого преобразования Хафа составит около 12% от эффективной производительности операции сложения и менее 2% от общей пиковой производительности сопроцессора. Приведенные оценки позволяют определить ключевые направления разработки алгоритма БПХ для СРСА. А именно, что достижение оптимальной реализации вычислительной процедуры практически бессмысленно, поскольку основной вклад в производительность вносят лишь обмены СРСА с внешней памятью. В частности, это означает, что:

• необязательно задействовать все 4 секции, производительность одной секции (1 Гопс) в 2 раза превышает пиковую производительность обменов с ОЗУ;

• совмещение обменов с вычислениями не приводит к заметному повышению производительности (доля вычислений составляет около 12% от общего объема работы);

• для повышения производительности БПХ необходимо сократить число обменов между ОЗУ и СРСА, иными словами – в алгоритме БПХ требуется выявить повторную используемость данных.

6. СОКРАЩЕНИЕ ЧИСЛА ОБМЕНОВ

В процедуре mergeHT на два входных среза $h0$ и $h1$ с общим числом столбцов n приходится n сумм, т.е. большинство столбцов участвует в двух суммах, причем с разными циклическими сдвигами. В случае, когда результирующий срез четного размера и $n0 = n1 = n/2$ (см. Листинг 1), каждая пара столбцов преобразуется в два новых столбца суммированием с разным циклическим сдвигом. В случае, когда результирующий срез нечетного размера, выделить такие пары не удается, но каждый столбец (кроме одного начального), участвует как минимум в двух суммах подряд (в порядке увеличения индекса).

Повторное использование столбцов позволяет сократить загрузку/выгрузку данных в локальную память сопроцессора СРСА в 1.5 раза: каждый столбец в рамках фиксированной операции объединения двух срезов мы загружаем ровно один раз. Пиковая теоретическая производительность при такой оптимизации возрастет до 800 Мопс, а оценка загрузки СРСА повысится до 18%.

По-существу, с точки зрения оценки производительности задача БПХ свелась к задаче копирования массивов в памяти. А производительность этой операции можно измерить штатными тестами из состава библиотеки цифровой обработки сигналов для СРСА. Тест производительности функции копирования комплексных векторов показывает, что достигаемая на практике производительность копирования составляет 1.88 Гбайт/сек при размере копируемых данных 32 Кбайта. Переводя полученные результаты в единицы, используемые здесь, получаем, что практически достижимая верхняя оценка производительности БПХ составляет 470 Мопс.

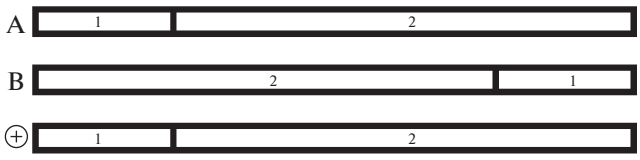

Рис. 3. Операция суммирования двух столбцов, суммируются части векторов A и B с одинаковым номером.

7. РЕАЛИЗАЦИЯ БПХ ДЛЯ СРСА

В данной работе авторы ограничились реализацией ключевой операции БПХ в одной секции СРСА, но подход можно обобщить на 4 секции. Предложенная реализация предполагает размещение векторов A, B и их суммы целиком в памяти секции, которая ограничена 16384 32-разрядными вещественными числами. Соответственно, максимальный допустимый размер столбца ограничен 5440 элементами, но при обобщении реализации на все 4 секции максимальный размер столбца может быть увеличен в 4 раза.

Поскольку шина между сумматорами и накристальной памятью СРСА имеет ширину 128 разрядов, максимальная производительность обменов с локальной памятью достигается при выравнивании данных по границе 16 байт. В силу этого сдвиги, кратные 4 элементам, осуществляются выбором соответствующего адреса в памяти секции, а меньшие сдвиги запрограммированы явно. Таким образом, для процедуры сложения векторов со сдвигом, применяющейся для суммирования частей 1 и 2 на рис. 3, было реализовано семь вычислительных процедур:

• сумма двух векторов A и B со сдвигом B на 0, 1, 2 и 3 элемента;

• сумма двух векторов A и B со сдвигом A и сдвигом записи в результирующий вектор на 1, 2 и 3 элемента.

Из уточнений в разделах 3 и 6 ясно, что поддержка изображений с размерами, не являющимися степенью двойки, усложняет программную реализацию. Для ее обеспечения потребовалось построение массива интервалов и два аналога процедуры mergeHT – для четного и нечетного числа столбцов. Массив интервалов строится за пренебрежимо малое время, а варианты mergeHT имеют одинаковый порядок вычислений, поэтому поддержка изображений произвольного размера не привела к ухудшению производительности БПХ.

Суммарный объем процедур – 200 строк кода на языке ассемблера СРСА, суммарное количество инструкций СРСА – 764. При этом с целью оптимизации выполнения кода при компиляции ассемблеру СРСА была задана глубина раскрутки циклов и повторения блоков, равная 8 инструкциям, т.е. большинство инструкций в итоговом коде размножены в 8 экземплярах. Оставшаяся логика алгоритма БПХ была реализована на языке программирования С, объем этого кода составляет около 400 строк.

В процессе создания кода он верифицировался и отлаживался на потактовом эмуляторе СРСА, входящим в состав кросс-библиотеки цифровой обработки сигналов. Производительность полученного кода измерялась на реальной аппаратуре – ПЭВМ на базе СнК 1890ВМ9Я под управлением ОС Debian GNU/Linux версии 8.

Измеренная производительность БПХ на изображении 5439×5439 составила 1.1 изображения/сек или 406 Мопс, т.е. 86% от полученной ранее верхней оценки производительности. Производительность ожидаемо возрастает с увеличением размера изображения. В частности, производительность обработки изображений 512 × 512 составляет 116 Мопс (49 изобр./сек), 1024 × 1024 – 191 Мопс (18 изобр./сек), 2048 × 2048 – 286 Мопс (6 изобр./сек), 4096×4096 – 378 Мопс (1.8 изобр./сек). Для неквадратных изображений производительность определяется размером столбца, поскольку именно с увеличением столбца изображения повышается эффективность загрузки СРСА и нивелируются накладные расходы: 512×32 – 112 Мопс (1365 изобр./сек), 512 × 256 – 118 Мопс (112 изобр./сек), 32×512 – 8.4 Мопс (57 изобр./сек), 256×512 – 64 Мопс (55 изобр./сек).

Для проверки оценок и, тем самым, подтверждения правильности принятых решений по выбору ключевых направлений разработки алгоритма БПХ для СРСА, была подготовлена специальная тестовая версия алгоритма. В этой специальной версии все вызовы вычислительных процедур удалены, т.е. измеряется производительность обменов между СРСА и ОЗУ. Производительность специальной версии на изображении 5439×5439 составила 458 Мопс. Это позволяет сделать следующие оценки:

• накладные расходы в реализации алгоритма составляют 2.5% от полученной ранее верхней оценки производительности;

• потери из-за несовмещения обменов с вычислениями составляют 11% при теоретически полученной оценке в 12%.

8. ЗАКЛЮЧЕНИЕ

В работе предложена новая реализация быстрого преобразования Хафа, учитывающая особенности архитектуры СРСА СнК 1890ВМ7Я. Как и у исходного рекурсивного алгоритма, вычислительная сложность предложенного варианта составляет $O(N^2 \log N)$. При этом столбцы изображения обрабатываются как векторы, а число векторных операций оценивается как $O(N \log N)$. Предложенная реализация полностью верифицирована на тестовых изображениях и путем сравнения с рекурсивной реализацией на языке MATLAB.

Ввиду слабой вычислительной емкости алгоритма БПХ загрузка СРСА не превосходит 18%. Показано, что основным фактором, определяющим производительность алгоритма, является пропускная способность ОЗУ. В силу этого ключевыми при оптимизации являются техники, опирающиеся на повторную используемость данных. Таким образом, использование знаний об особенностях архитектуры вычислителя позволило разработать эффективную реализацию БПХ. Выполненные теоретические и экспериментальные оценки производительности продемонстрировали небольшую долю накладных расходов.

9. БЛАГОДАРНОСТИ

Работа проводилась при частичной финансовой поддержке грантов РФФИ 18-29-26017 и 18-29-26027

СПИСОК ЛИТЕРАТУРЫ

1. *Brady M.L.* A fast discrete approximation algorithm for the Radon transform // SIAM Journal on Computing. 1998. V. 27. № 1. P. 91–99.
2. *Nikolaev D.P., Karpenko S.M., Nikolayev I.P.* Hough Transform: Underestimated Tool In The Computer Vision Field // Proceedings of 22nd European Conference on Modelling and Simulation. 2008. P. 238–243.
3. *Hough P.V.C.* Machine Analysis of Bubble Chamber Pictures // Conf. Proc. C. 1959. V. 590914. P. 554–558.
4. *Duda R.O., Hart P.E.* Use of the Hough Transformation to Detect Lines and Curves in Pictures // Commun. ACM. 1972. V. 15. № 1. P. 11–15.
5. *Асватов Е.Н., Ершов Е.И., Николаев Д.И.* Робастная ортогональная линейная регрессия для маломерных гистограмм // Сенсорные системы. 2017. Т. 31. № 4. С. 331–342.
6. *Nikolaev D.P., Nikolayev P.P.* Linear color segmentation and its implementation // Computer Vision and Image Understanding. 2004. V. 94. № 1. P. 115–139. Special Issue: Colour for Image Indexing and Retrieval.
7. *Kunina I.A., Gladilin S.A., Nikolaev D.P.* Blind compensation of radial distortion in a single image using fast Hough transform // Computer Optics. 2016. V. 40. № 3. P. 395–403.
8. A new approach to highway lane detection by using Hough transform technique / *Nur Shazwani Aminuddin, Masrullizam Mat Ibrahim, Nursabillilah Mohd Ali et al.* // Journal of Information and Communication Technology. 2017. V. 16. № 2. P. 244–260.
9. *Кунина И.А., Панфилова Е.И., Поволоцкий М.* Детектирование пешеходных переходов на изобра-

жениях дороги на основе метода динамического выравнивания временных рядов // Труды Института системного анализа Российской Академии наук (ИСА РАН). 2018. Т. 68. № S1. С. 23–31.

10. *Panfilova E.I., Shipitko O.S., Kunina I.A.* Fast Hough Transform-Based Road Markings Detection For Autonomous Vehicle // Proc. SPIE 11605, Thirteenth International Conference on Machine Vision (ICMV 2020). 2021. V. 11605. P. 671–680.
11. *Jahan R., Suman P., Singh D.K.* Lane detection using canny edge detection and hough transform on raspberry Pi // International Journal of Advanced Research in Computer Science. 2018. V. 9. № S2. P. 85–89.
12. Energy-Efficient Hardware Implementation of Road-Lane Detection Based on Hough Transform with Parallelized Voting Procedure and Local Maximum Algorithm / *J. Guan, F. An, X. Zhang et al.* // IEICE Transactions on Information and Systems. 2019. V. E102.D. № 6. P. 1171–1182.
13. *Thongpan Narathip, Rattanasiriwongwut Montean, Ketcham Mahasak*. Lane Detection Using Embedded System // International Journal of the Computer, the Internet and Management. 2020. V. 28. № 2. P. 46–51.
14. *Котов А.А., Коноваленко И.А., Николаев Д.П.* Прослеживание объектов в видеопотоке, оптимизированное с помощью быстрого преобразования Хафа // Информационные технологии и вычислительные системы. 2015. № 1. С. 56–68.
15. *Бочаров А.Д.* Метод линейной регрессии, устойчивый к экстремальным стационарным помехам // Сенсорные системы. 2020. Т. 34. № 1. С. 44–56.
16. *Green A.I., Schwartz R., Dodge J., Smith N.A., Etzioni O.* // Communications of the ACM. 2020. V. 63. № 12. P. 54–63.
17. *Tropin D.V., Ilyuhin S.A., Nikolaev D.P., Arlazarov V.V.* Approach for Document Detection by Contours and Contrasts. 2020. arXiv:2008.02615 [cs.CV].
18. *Bezmaternykh P.V., Nikolaev D.P.* A document skew detection method using fast Hough transform // Proc. SPIE 11433, Twelfth International Conference on Machine Vision (ICMV 2019). 2020. V. 11433. P. 132–137.
19. *Гайер А.В., Шешкус А.В.* Нейросетевая детекция верхней и базовой линий текстовой строки на изображении // XII мультиконференция по проблемам управления (МКПУ-2019). 2019. С. 53–58.
20. Slant rectification in Russian passport OCR system using fast Hough transform / *E. Limonova, P. Bezmaternykh, D. Nikolaev, V. Arlazarov* // Proc. SPIE 10341, Ninth International Conference on Machine Vision (ICMV 2016). 2017. V. 10341. P. 127–131.
21. *Martynov S.I., Bezmaternykh P.V.* Aztec core symbol detection method based on connected components extraction and contour signature analysis // Proc. SPIE 11433, Twelfth International Conference on Machine Vision (ICMV 2019). 2020. V. 11433. P. 27–34.
22. *Bulatov K.B., Chukalina M.V., Nikolaev D.P.* Fast X-ray sum calculation algorithm for computed tomography // Bulletin of the South Ural State University, Ser.: Mathematical Modelling Programming and Computer Software. 2020. V. 13. № 1. P. 95–106.
23. *Dolmatova A.V., Chukalina M.V., Nikolaev D.P.* Accelerated FBP for Computed Tomography Image Reconstruction // 2020 IEEE International Conference on Image Processing. 2020. P. 3030–3034.
24. Рентгеновский компьютерный томограф – новый инструмент в распознавании / *А. С. Ингачева, А. В. Шешкус, Т. С. Чернов и др.* // Труды Института системного анализа Российской академии наук (ИСА РАН). 2018. Т. 68. № S1. С. 90–99.
25. Monitored Reconstruction: Computed Tomography as an Anytime Algorithm / *K. Bulatov, M. Chukalina, A. Buzmakov et al.* // IEEE Access. 2020. V. 8. P. 110759–110774.
26. Vanishing point detection with direct and transposed fast Hough transform inside the neural network / *A. Sheshkus, A. Chirvonaya, D. Nikolaev, V. L. Arlazarov* // Computer Optics. 2020. V. 44. № 5. P. 737–745.
27. *Sheshkus A., Nikolaev D.* Houghencoder: Neural Network Architecture for Document Image Semantic Segmentation // 2020 IEEE International Conference on Image Processing. 2020. P. 1946–1950.
28. *Lin Y., Pintea S.L., van Gemert J.C.* Deep Hough-Transform Line Priors // Computer Vision – ECCV 2020. 2020. P. 323–340.
29. Deep Hough Transform for Semantic Line Detection / *Q. Han*, K. Zhao, J. Xu, M.-M. Cheng // Computer Vision – ECCV 2020. 2020. P. 249–265.
30. Line detection via a lightweight CNN with a Hough Layer / *L. Teplyakov, K. Kaymakov, E. Shvets, D. Nikolaev* // Proc. SPIE 11605, Thirteenth International Conference on Machine Vision (ICMV 2020). 2021. V. 11605. P. 376–385.
31. Методы распознавания и обработки изображений в процессе строительства нефтяных и газовых скважин / *С. А. Усилин, В. В. Арлазаров, Д. Н. Путинцев, И. А. Тарханов* // Информационные технологии и вычислительные системы. 2020. № 1. С. 12–24.
32. *Zhao H., Zhang Z.* Improving Neural Network Detection Accuracy of Electric Power Bushings in Infrared Images by Hough Transform // Sensors. 2020. V. 20. № 10. P. 1–16.
33. *Ershov E.I., Karpenko S.M.* Fast Hough Transform and approximation properties of dyadic patterns. 2017. arXiv: 1712.05615 [cs.CV].
34. *Ершов Е.И., Терехин А.И., Николаев Д.И.* Обобщение быстрого преобразования Хафа для трехмерных изображений // Информационные процессы. 2017. Т. 17. № 4. С. 294–308.
35. *Mehta D.P., Sahni S.* Handbook of Data Structures and Applications, Second Edition. Chapman & Hall/CRC, 2018.
36. ФГУ ФНЦ НИИСИ РАН. Микросистема интегральная 1890ВМ9Я. Указания по применению, 2016.
37. *Сударева О.Ю.* Эффективная реализация алгоритмов быстрого преобразования Фурье и свертки на микропроцессоре КОМДИВ128-РИО. М.: НИИСИ РАН, 2014.
38. *Райко Г.О., Мельканович В.С., Павловский Ю.А.* Технология программирования многопроцессорной обработки гидроакустических сигналов на вычислительных устройствах семейства “КОМДИВ” // Гидроакустика. 2014. № 20 (2). С. 85–92.