РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

АВТОМЕТРИЯ

2006, том 42, № 3

УДК [004.272.26 : 004.932] (043)

ВЕРТИКАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ В ОБРАБОТКЕ ИЗОБРАЖЕНИЙ НА МНОГОПРОЦЕССОРНЫХ ЭВМ *

Е. В. Русин

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск
E-mail: rev@ooi.sscc.ru

Рассмотрена экспериментальная библиотека параллельной обработки изображений PLVIP, построенная на основе вертикального представления данных. Приведены основные характеристики библиотеки (формат данных, организация вычислительного процесса, реализованные подпрограммы), а также примеры ее применения.

Введение. Гигантские объемы данных дистанционного зондирования Земли (информационные потоки до 128 Мбит/с при объеме одного изображения до 1 Гбайт, суточный объем принимаемой информации до 60 Гбайт), а также потребность их интерпретации в реальном времени (в задачах мониторинга лесных пожаров, наводнений и пр.) требуют высокой производительности ЭВМ, привлекаемых к их обработке. На сегодняшний день очевидно, что только повышением тактовой частоты вычислителя требуемой производительности достичь нереально, и многопроцессорная обработка является здесь безальтернативным средством получения результатов за требуемое время.

Существующие системы обработки изображений принадлежат, как правило, одному из классов: либо это универсальные и тщательно отлаженные библиотеки подпрограмм, ориентированные на ПК, такие как Intel IPP [1] или OpenCV [2], либо предназначенные для решения узкоспециальных задач программы для многопроцессорных суперЭВМ, либо еще более специализированные системы на базе оригинальной аппаратуры (например, матриц памяти с программируемой логикой для мелкозернистой обработки изображений). В то время как для многих областей науки, требующих трудоемких вычислений (численных методов решения задач механики, химической кинетики и др.), создано несколько крупных пакетов программ обработки изображений для суперЭВМ, фундаментальных библиотек подпрограмм на

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 05-07-90057) и Президиума РАН (интеграционный проект № 13.9).

базе многопроцессорных ЭВМ общего назначения в настоящее время не существует.

Предлагаемая работа посвящена описанию экспериментальной библиотеки параллельной обработки изображений PLVIP (Parallel Library for Vertical Image Processing), разработанной и реализованной в Лаборатории обработки изображений ИВМиМГ СО РАН. Алгоритмическим фундаментом при создании библиотеки стал принцип вертикальной обработки, обеспечивающий эффективность обработки малоразрядных данных, например полутоновых малоградационных изображений. Библиотека функционирует на многопроцессорных ЭВМ Сибирского суперкомпьютерного центра СО РАН (32-процессорном Linux-кластере МВС-1000/М и 8-процессорном SMP-сервере RM600-E30).

В данной работе после краткого описания метода вертикальной обработки рассматриваются основные характеристики библиотеки PLVIP (поддерживаемые форматы данных, организация вычислений и назначение подпрограмм). Применение разработанных программных средств иллюстрируется примером решения классической задачи цифровой картографии – восстановления карты высот по заданному набору горизонталей.

Вертикальная обработка. В 70-е годы прошлого века для поддержки решения задач, допускающих массовую обработку информации, в том числе задач обработки изображений, были созданы специализированные мелкозернистые SIMD-комплексы, состоящие из очень большого (десятки тысяч) числа синхронно действующих одноразрядных процессорных элементов с собственной памятью, объединенных соединительной сетью [3]. Основной особенностью таких ЭВМ стал вертикальный, или пословно-параллельный поразрядно-последовательный, подход к обработке данных (вертикальная обработка (ВО)) [4].

Схематично принцип ВО иллюстрирует рис. 1. В отличие от традиционной обработки, когда элементы массива поступают в процессор по очереди и каждый элемент обрабатывается в процессоре целиком, при ВО в процессор одновременно поступают одноименные, т. е. одного старшинства, разряды сразу нескольких данных.

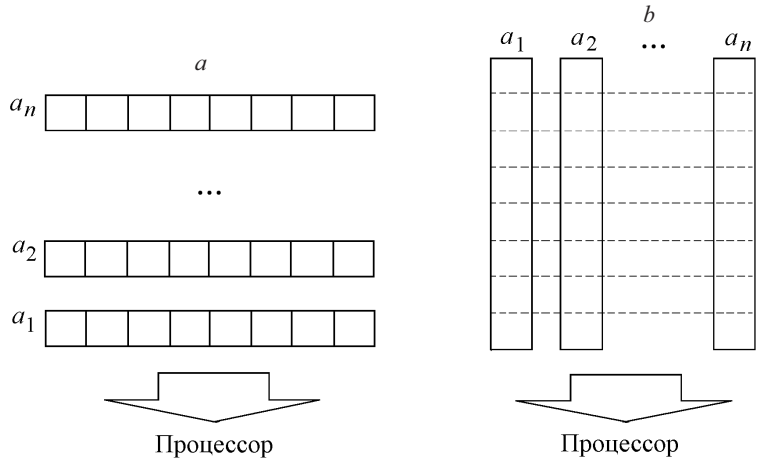

Рис. 1. Обработка массива информации: традиционным (горизонтальным) (*a*) и вертикальным (*b*) способами

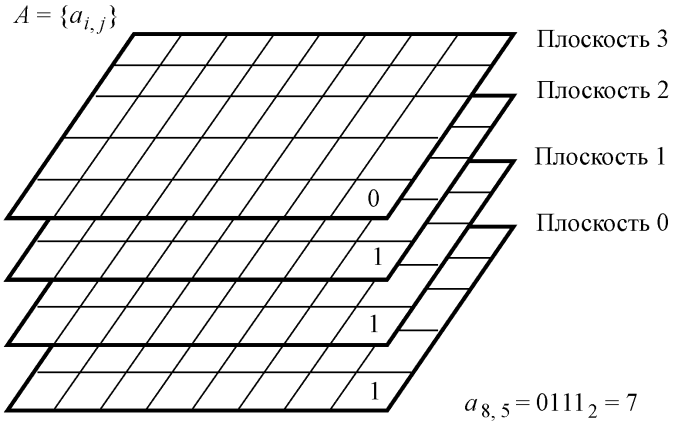

Рис. 2. 16-цветное изображение A, состоящее из четырех разрядных плоскостей (плоскость 0 содержит младшие разряды значений пикселей, плоскость 3 – старшие); значение пикселя (8, 5) формируется из значений соответствующих ему битов разрядных плоскостей A

Набор значений одноименных разрядов пикселей изображения называется разрядной плоскостью изображения. Именно разрядные плоскости являются обрабатываемыми единицами при вертикальной обработке изображений (ВОИ), значение пикселя является композицией значений соответствующих битов разрядных плоскостей изображения (рис. 2).

Базой алгоритмики ВОИ является группа операций с разрядными плоскостями:

– побитовые логические операции между плоскостями;

– сдвиги плоскостей в горизонтальном и вертикальном направлениях;

– вычисление «массы» плоскости, т. е. числа единичных битов в ней;

– проверка наличия хотя бы одного единичного (нулевого) бита в плоскости;

– сравнение плоскостей на равенство.

Модель данных, используемая при ВО, не нашла широкого распространения вне спецпроцессоров, в частности неисследованными на сегодняшний день являются вопросы реализации вертикальной обработки изображений в ЭВМ общего назначения. Между тем для исследования этих вопросов существует ряд предпосылок.

1. Производительность поразрядных вычислений растет с уменьшением разрядности обрабатываемых данных, что позволяет сделать более эффективной обработку малоразрядных данных, например полутоновых малоградационных изображений.

2. При ВО не существует ограничений на размер обрабатываемых данных, что дает возможность легко выполнять вычисления с произвольной точностью.

3. Алгоритмическая база ВОИ эффективно реализуется на ЭВМ общего назначения побитовыми логическими операциями между машинными словами и их сдвигами; современные 64-разрядные микропроцессоры позволяют одновременно обрабатывать 64 пикселя изображения.

4. Явный параллелизм ВО дает возможность использовать для ее реализации универсальные многопроцессорные ЭВМ.

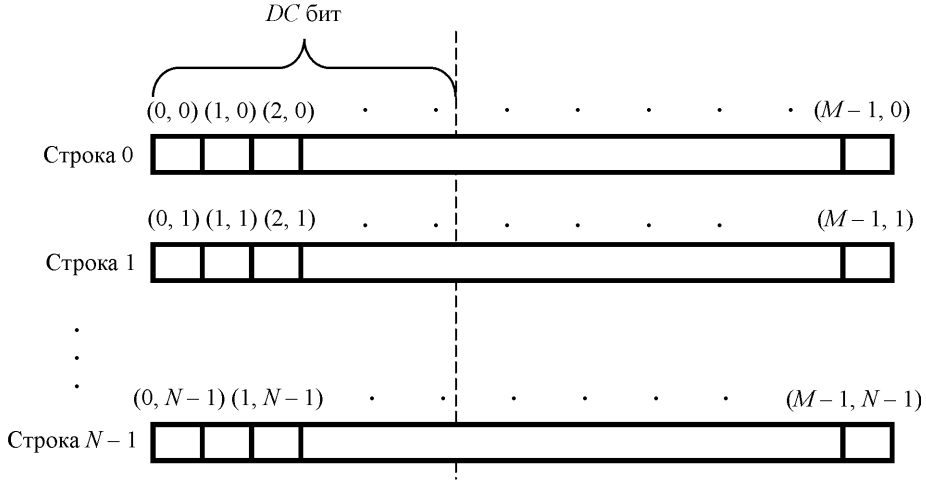

Рис. 3. Построчно-непрерывный формат битовой плоскости

Данные соображения стали основанием для создания на базе принципа ВО библиотеки параллельной обработки изображений PLVIP, функционирующей на широко распространенных многопроцессорных MIMD-ЭВМ с традиционной архитектурой.

Библиотека PLVIP. Библиотека PLVIP [5] реализована как набор С-подпрограмм. В качестве средства реализации выбран язык С и интерфейс параллельного программирования MPI, что обеспечивает как возможность низкоуровневых манипуляций с разрядными плоскостями, так и переносимость исходного кода библиотеки на подавляющее большинство современных многопроцессорных архитектур.

Формат данных. Ключевой задачей при переносе концепции ВО на традиционные вычислительные средства является разработка формата разрядной плоскости. Невозможность реализации полноценной двумерной топологии на линейном адресном пространстве традиционной ЭВМ привела к компромиссному варианту, аналогичному способу отображения видеопамяти на ОЗУ в некоторых графических режимах видеоадаптера VGA: разрядная плоскость изображения размером $M \times N$ пикселей представляется набором из N битовых строк, образуемых последовательно расположенными в памяти машинными словами. При этом число слов в строке равно M/DC, где DC – разрядность процессора (полагаем M кратным DC). Для того чтобы отношение соседства поддерживалось сдвигами машинных слов, соответствие между битами строк и пикселями изображения определяется так: i-й по старшинству (начиная со старшего) бит j-го по порядку (начиная с младших адресов) слова k-й строки разрядной плоскости соответствует пикселю с координатами $(jDC + i, k)$ (рис. 3).

На данной основе строятся представления многоградационных изображений, целочисленных и в формате с фиксированной запятой, как наборы разрядных плоскостей.

Организация вычислений. В качестве основного принципа распараллеливания вычислений был выбран принцип декомпозиции области: обрабатываемые разрядные плоскости (а следовательно, и изображения, образуемые ими) разрезаются на непрерывные непересекающиеся горизонтальные по-

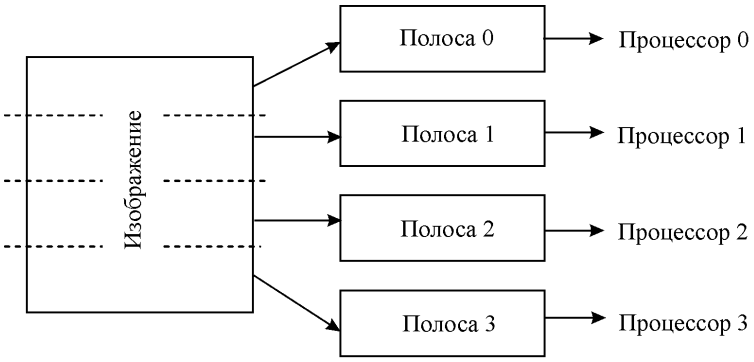

Рис. 4. Принцип распараллеливания вычислений, реализованный в библиотеке PLVIP

лосы, число которых равно числу процессоров-обработчиков. Далее полосы распределяются между процессорами (рис. 4).

Данный принцип обеспечивает эффективное распараллеливание логических операций с плоскостями и горизонтальных сдвигов плоскостей. Вместе с тем вертикальные сдвиги, вычисление массы плоскости и сравнение плоскостей требуют межпроцессорного обмена при таком способе распараллеливания. И если зависимость параллельных процессов при выполнении операций сравнения плоскостей и вычисления массы в практических задачах часто может быть существенно ослаблена путем использования на каждом процессоре результатов сужения этих операций на полосу, соответствующую процессору, то интенсивное использование алгоритмом вертикальных сдвигов плоскости делает его непригодным для такого распараллеливания. Ввиду этого для обеспечения альтернативных способов распараллеливания большинство операций реализовано в библиотеке PLVIP как для разрезанных на полосы изображений, так и для изображений, целиком находящихся на одном процессоре.

В качестве топологии вычислений выбрана «звезда»: операции ввода–вывода и преобразования форматов изображений осуществляются одним, «корневым» процессором; остальные процессоры получают данные от корня, обрабатывают их и в случае потребности возвращают результат обратно. Выбор такой топологии является следствием особенности файловой системы ЭВМ МВС-1000/М.

Подпрограммы библиотеки. Библиотека включает:

– подпрограммы инициализации и завершения работы библиотеки;

– подпрограммы ввода–вывода в форматах PCX, BMP и внутреннем формате библиотеки;

– подпрограммы разрезания изображений между процессорами и сборку изображений на корне из полос;

– базовые подпрограммы для бинарных изображений (копирование, побитовые логические операции, сдвиги, проверка равенства и вычисление массы);

– общие подпрограммы для многоградационных изображений (копирование, преобразование вертикального формата в традиционный и обратно, задание–получение значения пикселя и т. п.);

Таблица 1

Алгоритм	Число процессоров				
	2	4	8	16	24
Попиксельное умножение	1,6	4,4	10,5	18,0	24,4
Скалярное произведение	2,0	3,8	7,4	14,4	19,9
Построение гистограммы	2,2	3,7	10,0	18,5	21,7
Алгоритм Зонга – Суня	1,7	3,6	6,1	8,8	10,3
Построение линии уровня значений пикселей	1,7	3,3	7,6	10,0	20,0
Построение поля евклидовых расстояний	2,0	4,7	7,5	15,5	20,0

– подпрограммы арифметики многоградационных изображений (попиксельное сложение, вычитание, умножение, деление, вычисление суммы значений пикселей, скалярное произведение изображений);

– подпрограммы обработки полутоновых многоградационных изображений (построение гистограммы, профилей, линии уровня значений пикселей, утоньшение по Зонгу – Суню [6] и построение поля евклидовых расстояний).

Табл. 1 содержит значения ускорения выполнения некоторых подпрограмм библиотеки PLVIP на 2, 4, 8, 16, 24 процессорах по отношению к однопроцессорному выполнению (размер тестового изображения 640×640 пикселей, вычисления выполнялись на ЭВМ МВС-1000/М).

В некоторых случаях значение ускорения превышает число исполняющих процессоров N, что объясняется двумя факторами:

– с ростом N объем данных, которым оперирует один процессор, уменьшается, и, начиная с некоторого $N = N_*$, все данные попадают в кэш процессора, выборка из которого значительно быстрее, чем из главной памяти;

– в ряде алгоритмов применена оптимизация, которая позволяет сократить объем вычислений, опираясь на глобальные свойства обрабатываемой полосы изображения (отсутствие единичных (нулевых) битов в полосе плоскости, совпадение соответствующих полос двух плоскостей и т. д.); с ростом N и соответственно с уменьшением размера обрабатываемой отдельным процессором полосы эффективность такой оптимизации растет (см. разд. 4).

4. Оптимизация алгоритмов вертикальной обработки изображений. Поразрядный характер вертикальной обработки приводит к прямой зависимости времени выполнения алгоритма от разрядности значения пикселя обрабатываемого изображения. В ряде случаев ослабить эту зависимость позволяет оптимизация вычислений ветвлением типа «если i-я разрядная плоскость изображения заполнена исключительно нулями (единицами), выполнить A; в противном случае – B».

Для примера рассмотрим алгоритм умножения двух двоичных чисел a и b «столбиком». Представив

$$a = a_{N-1}2^{N-1} + a_{N-2}2^{N-2} + \ldots + a_0,$$

где $a_i \in \{0,1\}$, получим

$$ab = a_{N-1}2^{N-1}b + a_{N-2}2^{N-2}b + \ldots + a_0 b =$$

$$= a_{N-1}\mathrm{SL}(b, N-1) + a_{N-2}\mathrm{SL}(b, N-2) + \ldots + a_0 b$$

(здесь $\mathrm{SL}(x, y)$ – сдвиг влево числа x на y двоичных разрядов). Тогда вычисления выполняются по следующему алгоритму:

$$\begin{aligned}
&S_0 = 0;\\
&S_1 = S_0 + a_0 b;\\
&S_2 = S_1 + a_1 \mathrm{SL}(b,1);\\
&\ldots\\
&S_i = S_{i-1} + a_{i-1}\mathrm{SL}(b, i-1);\\
&\ldots\\
&S_{N-1} = S_{N-1} + a_{N-1}\mathrm{SL}(b, N-1);\\
&ab = S_{N-1}.
\end{aligned} \qquad (1)$$

Если допустить, что временные затраты на определение истинности утверждения «$a_i = 0$» существенно меньше, чем на сдвиг многоразрядного числа и сложение двух многоразрядных чисел, то оказывается возможной оптимизация алгоритма умножения по схеме:

$$\begin{aligned}
&S_0 = 0;\\
&\ldots\\
&S_i = \begin{cases} S_{i-1} + a_{i-1}\mathrm{SL}(b, i-1), & \text{если } a_{i-1} \neq 0;\\ S_{i-1} & \text{иначе,} \end{cases}\\
&\ldots\\
&ab = S_{N-1}.
\end{aligned} \qquad (2)$$

Если переписать вычислительные процессы (1) и (2) для изображений, то условию $a_{i-1} \neq 0$ процесса (2) будет соответствовать условие «i-я разрядная плоскость изображения A содержит только нулевые биты». Проверка данного условия значительно менее трудозатратна, чем сложение двух многоразрядных изображений, поэтому такая оптимизация вычислений выглядит перспективной.

Справедливость данных рассуждений иллюстрируется результатами оптимизации алгоритмов построения гистограммы и вертикального профиля изображения. Табл. 2 представляет временные результаты (в секундах) применения неоптимизированного и оптимизированного алгоритмов построения гистограммы и вертикального профиля к двум одинаковым изображениям I_1 и I_2 размером 1600×1400 пикселей, которые принимают 231 различное значение в диапазоне от 0 до 255. Для представления I_1 используется 8 разрядных плоскостей (256 градаций серого), а для I_2 – 16 плоскостей

Таблица 2

Алгоритм	Изображение	
	I_1	I_2
Гистограмма (неоптимизированный алгоритм)	0,3700	93,0000
Гистограмма (оптимизированный алгоритм)	0,3000	0,3200
Вертикальный профиль (неоптимизированный алгоритм)	0,0072	0,0178
Вертикальный профиль (оптимизированный алгоритм)	0,0072	0,0103

(65536 градаций серого): старшие восемь разрядных плоскостей изображения I_2 заполнены нулями.

Временная сложность неоптимизированного алгоритма построения гистограммы, т. е. время его исполнения как функция от разрядности значения пикселя L, составляет $O(2^L)$ [7]. Как следует из табл. 2, время выполнения оптимизированного алгоритма оказывается почти независимым от L. Оптимизация алгоритма построения вертикального профиля, временная сложность которого составляет $O(L)$, оказывается менее эффективной из-за сопоставимости вычислительных затрат на оптимизацию и затрат непосредственно на вычисления. Тем не менее рост времени исполнения оптимизированного алгоритма с увеличением разрядности пикселя оказывается медленней линейного.

Такая оптимизация дает еще больший выигрыш при распараллеливании декомпозицией области: теперь каждый процессор оперирует частью изображения, причем число различных значений, принимаемых пикселями подызображения, в большинстве случаев будет заметно меньше числа различных значений пикселей целого изображения.

5. Решение задачи восстановления карты высот по набору горизонталей. *Постановка задачи.* С помощью библиотеки PLVIP была решена задача восстановления растровой карты высот рельефа земной поверхности по заданному набору горизонталей, рассматриваемая в следующей постановке [8]:

1) пусть дан прямоугольный участок B растровой плоскости с квадратными элементами (пикселями);

2) горизонталью уровня H называется множество всех пикселей B, значение высоты земной поверхности в которых равно H;

3) пусть на B дан набор горизонталей $C_1, C_2, \ldots, C_S$ уровней $L_1, L_2, \ldots, L_S$ соответственно и $L_i < L_j$ при $i < j$;

4) заданный набор горизонталей отражает непрерывный характер земной поверхности, т. е. не найдется двух горизонталей C_i и C_j, $j > i+1$, не разделенных горизонталями $C_{i+1}, C_{i+2}, \ldots, C_{j-1}$.

На основании 1–4 требуется аппроксимировать значение высоты земной поверхности на множестве $B / \bigcup_i C_i$.

В работе [9] показано, что четвертое условие, вообще говоря, можно ослабить, допустив возможность пересечения горизонталей разных уровней. Однако соответствующие изменения в алгоритме не оказывают сущест-

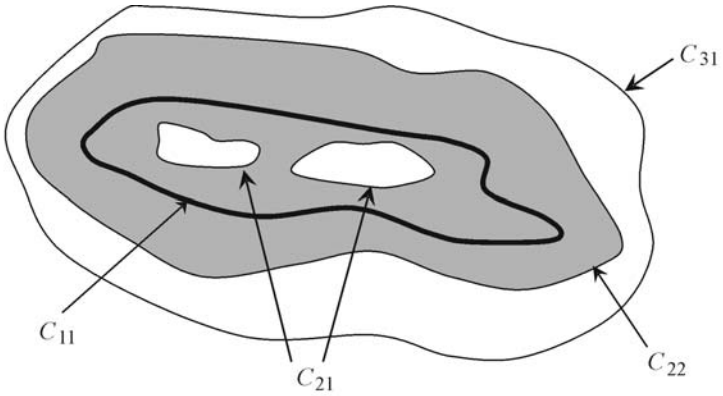

Рис. 5. Сегменты горизонталей C_1, C_2 и C_3; серым цветом выделена область $\mathrm{dep}(C_{11})$

венного влияния на его распараллеливание, поэтому все дальнейшие рассуждения будут относиться к более простой постановке 1–4.

Алгоритм решения. Исследуемый алгоритм решения задачи подробно изложен в [9]. Не вдаваясь в подробности, выделим основные этапы этого алгоритма.

1. Каждая горизонталь C_i разбивается на набор непересекающихся подмножеств пикселей C_{ij}, $j = 1, \ldots, J(i)$ (сегментов), причем любые два различных сегмента одной горизонтали C_i отделены друг от друга множеством $D_i = \bigcup_{k \neq i} C_k$.

2. Для каждого сегмента C_{ij} находится область зависимости $\mathrm{dep}(C_{ij})$ – множество пикселей, не отделенных от C_{ij} множеством D_i (рис. 5). В $\mathrm{dep}(C_{ij})$ строится поле Dist_{ij} евклидовых расстояний до C_{ij} и поле Hight_{ij} уровня сегмента, значение которого в каждом пикселе $\mathrm{dep}(C_{ij})$ равно L_i.

3. Из построенных в предыдущем пункте локальных полей составляются четыре глобальных поля: Dist_0 и Dist_1, которые являются композициями полей расстояний до сегментов горизонталей с четными и нечетными номерами соответственно (операция композиции корректна, так как области зависимости сегментов несоседних горизонталей, равно как и области зависимости различных сегментов одной горизонтали, не пересекаются), а также Hight_0 и Hight_1 – аналогичные композиции полей уровней сегментов.

4. Поле аппроксимирующих высоту значений H находится как результат массовой арифметической операции, примененной к полям, полученным в предыдущем пункте:

$$H = \frac{\mathrm{Dist}_0 \mathrm{Hight}_1 + \mathrm{Dist}_1 \mathrm{Hight}_0}{\mathrm{Dist}_0 + \mathrm{Dist}_1}.$$

Результат работы алгоритма приведен на рис. 6.

Распараллеливание алгоритма с помощью библиотеки PLVIP. Основная вычислительная нагрузка при исполнении алгоритма ложится на две его части: выделение сегментов горизонталей и построение для них полей расстояний (массовая арифметическая операция занимает малую часть времени работы алгоритма и хорошо распараллеливается по данным). При этом лишь

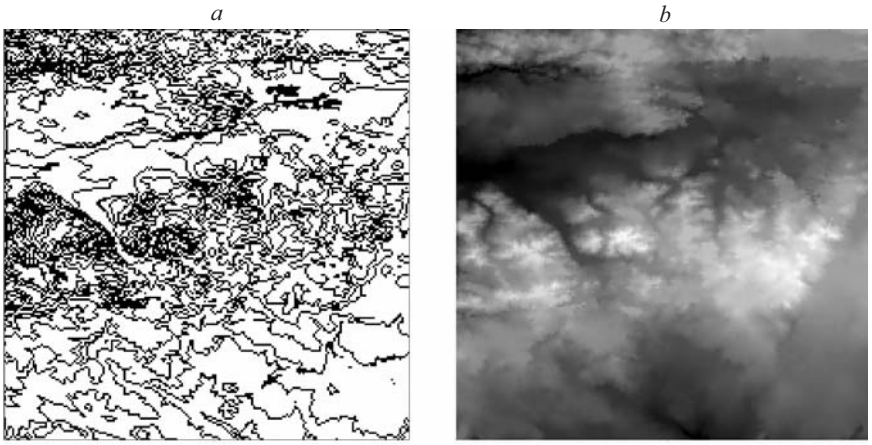

Рис. 6. Применение алгоритма аппроксимации: набор из 15 горизонталей (*a*), результат аппроксимации исходной растровой карты (*b*)

построение поля расстояний хорошо распараллеливается по данным (см. табл. 1). Операция же выделения сегмента и построения области его зависимости состоит из итерационной последовательности сдвигов бинарного изображения горизонтали; кроме того, условие завершения выделения сегмента есть предикат, относящийся к некоторому изображению в целом, причем это изображение изменяется от итерации к итерации.

Рассмотрим три подхода к распараллеливанию алгоритма, реализованные с помощью PLVIP.

Алгоритм A_1 – базовый способ распараллеливания для библиотеки PLVIP: каждое участвующее в вычислениях изображение разрезается на горизонтальные полосы, которые распределяются между процессорами. Таким образом, вертикальный сдвиг изображения приводит к передаче информации через границы соседних полос, а проверка условия завершения выделения сегмента – к синхронизации работы всех обрабатывающих процессоров. Далее для параллельного вычисления поля расстояний до сегмента каждому процессору требуется изображение полученного сегмента целиком, что вызывает необходимость сборки этого изображения на всех процессорах. Очевидным недостатком данного подхода являются интенсивные межпроцессорные обмены.

Алгоритм A_2 – распараллеливание по горизонталям: каждый процессор обрабатывает (т. е. выделяет сегменты и вычисляет для них области зависимости и поля расстояний и уровней) свое подмножество горизонталей, причем горизонталь целиком обрабатывается одним процессором. После этого построенные локальные поля расстояний до сегментов и уровней сегментов собираются на корневом процессоре, где из них формируются поля $\mathrm{Dist}_{0(1)}$ и $\mathrm{Hight}_{0(1)}$, которые представляют собой изображения в формате с фиксированной запятой, разрезаются далее на полосы и распределяются между процессорами для выполнения финальной арифметической операции. Преимуществом данного подхода является малая зависимость параллельных процессов (синхронизация и коммуникации требуются только при сборе полученных данных), недостатком – ограниченное числом горизонталей число исполняющих процессоров и возможная их неравномерная загрузка, которая

зависит от количества и сложности горизонталей, обрабатываемых процессором.

Алгоритм A_3 – распараллеливание по горизонталям с балансировкой нагрузки: выделение сегментов из каждой горизонтали выполняется целиком одним процессором. Процессоры, которым не досталось горизонтали или которые уже обработали свои горизонтали, по запросу получают выделенные сегменты и вычисляют для них поля расстояний. При отсутствии запроса на данный сегмент его обработку выполняет выделивший процессор. Обработка запросов выполняется корневым процессором, который освобождается от вычислений. Данный подход представляется самым гибким из перечисленных, здесь отсутствует жесткая синхронизация процессов, как в A_1, и достигается более равномерная загрузка процессоров, чем в A_2.

Вычислительные эксперименты были выполнены на МВС-1000/М над набором из 15 горизонталей на растровой карте размером 1600×1400 пикселей (общее число сегментов горизонталей равно 29951). Результаты экспериментов сведены в табл. 3.

В таблице для каждого алгоритма A_i приведено время выполнения T_{A_i} и ускорение по отношению к однопроцессорному исполнению Ac_{A_i} для различного числа исполняющих процессоров. Как видно, только для малого числа процессоров (2–3) скорость выполнения алгоритма A_3 уступает другим вариантам.

Подробному анализу участков, на которых происходит потеря эффективности всех трех алгоритмов, посвящена работа [10]. Здесь же отметим, что результаты экспериментов подтверждают правильность выбранной страте-

Таблица 3

Число процессоров	Алгоритмы					
	A_1		A_2		A_3	
	T_{A_1}	Ac_{A_1}	T_{A_2}	Ac_{A_2}	T_{A_3}	Ac_{A_3}
2	1136	2,27	1270	2,03	1943	1,33
3	911	2,83	965	2,67	1004	2,57
4	882	2,92	860	3,00	787	3,27
5	909	2,83	690	3,73	584	4,41
7	785	3,28	507	5,08	436	5,91
9	804	3,20	424	6,08	364	7,08
11	780	3,30	429	6,00	311	8,28
13	866	2,97	323	7,98	262	9,83
15	833	3,09	246	10,47	236	10,92
20	935	2,76	–	–	220	11,71
24	–	–	–	–	211	12,21

гии реализации библиотеки PLVIP – обеспечивать альтернативные базовому способы распараллеливания вычислений.

Заключение. Несмотря на серьезные ограничения (однородность вычислительного алгоритма, рост трудоемкости с увеличением разрядности обрабатываемых данных и др.), накладываемые вертикальным представлением данных, создание библиотеки PLVIP явилось важным шагом в понимании принципов реализации систем обработки изображений на многопроцессорных ЭВМ традиционной архитектуры. На основании полученного опыта в Лаборатории обработки изображений ИВМиМГ СО РАН начинается работа по созданию универсальной высокопроизводительной библиотеки обработки изображений на многопроцессорных ЭВМ Сибирского суперкомпьютерного центра СО РАН. Важными принципами ее создания станут многоформатность (поддержка как вертикального, так и традиционного форматов изображений) и многообразие методов распараллеливания (одно изображение на одном процессоре, разрезание на полосы, разрезание на полосы с перекрытием и т. д.). Кроме базовых алгоритмов обработки изображений, в библиотеке будут реализованы также оригинальные алгоритмы, созданные в лаборатории и ориентированные на обработку аэрокосмоснимков.

СПИСОК ЛИТЕРАТУРЫ

1. **Intel IPP** // http://intel.com/software/products/ipp/
2. **Open CV** // http://sourceforge.net/projects/opencvlibrary/
3. **Potter J. L., Meilander W. C.** Array processor supercomputers // Proc. IEEE. 1989. **77**, N 12. P. 1896.
4. **Shooman W.** Parallel computing with vertical data // Proc. EJCC 1960. **18**. P. 111.
5. **PLVIP** – библиотека подпрограмм параллельной обработки изображений: руководство программиста // http://loi.sscc.ru/lab/RFFI05/ru/PLVIP/main.htm
6. **Zhang T. Y., Suen S.Y.** A fast parallel algorithm for thinning digital patterns // Commun. ACM. 1984. **27**, N 3. P. 236.
7. **Reeves A. P.** On efficient global information extraction methods for parallel processors // Comput. Graph. and Image Process. 1980. **14**, N 2. P. 159.
8. **Русин Е. В.** Параллельный алгоритм аппроксимации матрицы высот земной поверхности по заданным горизонталям // Тр. Междунар. конф. «Математические методы в геофизике». Новосибирск: ИВМиМГ СО РАН, 2003. Ч. 2. С. 612.
9. **Kim P. A., Pyatkin V. P., Rusin E. V.** Three massively parallel algorithms for solving computational geometry problems by using euclidean distance transform // Pattern Recogn. and Image Analysis. 2004. **14**, N 2. P. 267.
10. **Русин Е. В.** О распараллеливании одного алгоритма восстановления растровой карты земной поверхности по заданным горизонталям // Тр. Междунар. конф. по вычислительной математике (МКВМ-2004). Новосибирск: ИВМиМГ СО РАН, 2004. Ч. 1. С. 197.

Поступила в редакцию 27 октября 2005 г.