ОБРАБОТКА ИЗОБРАЖЕНИЙ, РАСПОЗНАВАНИЕ ОБРАЗОВ

МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННОЙ СТРУКТУРЫ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Попов С.Б.
Учреждение Российской академии наук Институт систем обработки изображений РАН

Аннотация

Предложен подход к формированию моделей, описывающих информационную структуру параллельных программ обработки изображений. В качестве основы формальной модели параллельной обработки изображений выбрана теория взаимодействующих последовательных процессов, которая позволяет формировать иерархическое описание информационной структуры алгоритма, необходимое при построении эффективного отображения технологий обработки изображений на архитектуру распределенных систем.

Ключевые слова: информационная структура алгоритма, алгебра CSP, параллельная обработка изображений, модель изображения, декомпозиция изображений.

Введение

Одной из важнейших проблем использования вычислительной техники является «отображение задач вычислительной математики на архитектуру вычислительных систем». Эта проблема была обозначена академиком Г.И. Марчуком как фундаментальное научное направление, кратко называемое «проблемой отображения» [1].

В настоящее время наиболее актуальным представляется решение проблемы отображения вычислительных задач на параллельную архитектуру вычислительных систем, поскольку основным направлением повышения эффективности использования вычислительных средств является использование параллельных методов организации вычислений.

Ключевым моментом в решении проблемы отображения является анализ информационной структуры алгоритмов. Причем в работах В.В. Воеводина [2] была выдвинута гипотеза о том, что в конкретных вычислительных областях типовых информационных структур немного. Разработка таких информационных моделей алгоритмов в конкретных прикладных областях является актуальной задачей, поскольку открывает новые подходы к разработке специализированных программных средств, эффективно решающих задачи определенного класса на распределенных вычислительных системах, и спецпроцессоров, реализующих быстрое выполнение целевых алгоритмов.

Основной способ решения задачи отображения – математически эквивалентные преобразования модели (описания) информационной структуры алгоритма, в процессе которых формируются возможные варианты параллельной организации вычислений [2]. В процессе преобразования информационной модели алгоритма изменяется не только порядок вычислений, преобразованию подвергаются и структуры данных: выполняется декомпозиция данных на распределяемые, дублируемые и локальные, выявляются способы распределения, которые минимизируют коммуникационные издержки в процессе вычислений. Сформированные таким образом варианты параллельного решения задачи анализируются с точки зрения эффективности их реализации на некотором множестве возможных архитектур вычислительных систем.

Указанный подход к решению проблемы отображения предъявляет следующие основные требования к моделям, описывающим информационную структуру алгоритма.

Модель должна отражать все аспекты параллелизма, которые могут использоваться при отображении алгоритма на параллельную архитектуру вычислительной системы – параллелизм по данным и параллелизм по вычислениям. Параллелизм по вычислениям чаще всего описывает возможности одновременного выполнения основных внутренних операций, составляющих макроструктуру алгоритма. Параллелизм по данным формирует, как правило, наборы более низкоуровневых операций над элементами данных, которые могут выполняться независимо. Данные аспекты параллелизма достаточно тесно связаны и образуют своего рода иерархию: на верхнем уровне – независимые макрооперации (параллелизм по вычислениям), часть из которых состоит из параллельно выполняемых преобразований данных (параллелизм по данным).

Модель должна описывать распределение и перемещение данных, формируемое при параллельной организации вычислений.

Модель должна иметь средства оценки времени и других характеристик эффективности реализации каждого из вариантов размещения данных и организации вычислений, что влечет за собой необходимость адекватного и конструктивного расширения моделей информационной структуры алгоритма в направлении учета времени решения описываемой ими задачи на некотором множестве программно-аппаратных средств с параллельной архитектурой. Данное расширение информационной модели, как правило, связано с разработкой или выбором аналитической предиктивной модели, описывающей ха-

рактеристики производительности используемых программно-аппаратных средств целевой вычислительной системы, и методами достоверного оценивания параметров этой модели в различных режимах функционирования.

Основные подходы к построению моделей обработки изображений

Рассмотрим проблему отображения задач в области обработки изображений на архитектуру вычислительных систем, использующих различные типы параллелизма.

Рассмотрение объекта исследования – вычислительных задач обработки изображений, показало, что эти задачи имеют следующую присущую им особенность – большинство задач может быть решено путем последовательного применения к обрабатываемым изображениям некоторого набора типовых операций обработки [3]. Именно эту особенность эксплуатирует большинство систем обработки изображений (СОИз) общего назначения, т.е. подобные системы создаются не под конкретную технологию обработки, а для решения широкого спектра задач обработки изображений. В процессе формирования некоторой технологии обработки изображений пользователь оперирует вполне определенным набором реализованных в системе типовых операций обработки. Анализ задач обработки изображений показывет, что основных информационных структур, используемых при реализации этих операций, немного. Пример типологии операций низкоуровневой обработки изображений, которая совпадает с укрупненным разделением по способу организации информационной структуры, приведен, например, в [4] и включает в себя:

- поэлементную обработку (PO – point operators),
- обработку в локальной окрестности (LNO – local neighborhood operators) или локальную обработку скользящим окном,
- глобальные операции обработки (GO – global operators), как правило, основанные на двумерном преобразовании Фурье или другом подобном преобразовании,
- глобальные операции редукции (RO) или операции вычисления параметров изображения, в т.ч. статистических,
- геометрические преобразования.

Выбор методологии формального описания информационной структуры алгоритмов и программ при разработке информационных моделей технологий обработки изображений и решения с их помощью проблемы отображения должен учитывать иерархичность представления элементов модели. На различных уровнях абстракции модель должна отображать только те характеристики, которые являются существенными, характерными для данного уровня. Должны существовать различные уровни детальности описания модели для каждого уровня абстракции.

Для обработки изображений таковыми могут являться уровень описания технологии в целом; уровень модулей (блоков) в программе, реализующих отдельные алгоритмы или этапы технологии; информационная структура алгоритма, реализованная в отдельном модуле. На каждом уровне необходим свой алфавит модели информационной структуры системы на данном конкретном уровне абстракции.

Процесс формирования информационной модели задачи обработки изображений начнем с описания верхнего уровня представления решаемой пользователем задачи.

В общем виде процесс решения задачи обработки изображений предполагает либо изначальное указание последовательности некоторого числа шагов решения, либо возможность декомпозиции метода решения на составляющие части, реализуемые в виде низкоуровневых операций. На каждом шаге решения обрабатываются и/или вырабатываются элементы решения.

Формально этот процесс можно представить следующим образом.

На I-ом шаге формируются с помощью преобразования Φ^I необходимые элементы решения, представляемые, в общем случае, некоторым множеством параметров $\{\mathbf{r}\}^I$ и множеством изображений $\{\mathbf{a}\}^I$

$$\left\{\{\mathbf{a}\}^I,\{\mathbf{r}\}^I\right\} = \Phi^I\left(\{\mathbf{a}\},\{\mathbf{r}\},\mathbf{q}\right),$$

$$\{\mathbf{a}\} \subset \bigcup_{i=0}^{I-1}\{\mathbf{a}\}^i,\ \{\mathbf{r}\} \subset \bigcup_{i=0}^{I-1}\{\mathbf{r}\}^i$$

где $\{\mathbf{a}\}$, $\{\mathbf{r}\}$, $\mathbf{q}$ – множества входных изображений, входных параметров и вектор собственных параметров преобразования Φ^I соответственно. Любое из этих множеств на некотором конкретном шаге может быть пустым. $\{\mathbf{a}\}^0$, $\{\mathbf{r}\}^0$ – исходные множества изображений и параметров для рассматриваемой задачи.

Как правило, преобразование формирует только одно изображение или только один вектор параметров (или даже скаляр). В качестве входных параметров так же используются одно (реже два) изображение и некоторый набор параметров операции, часть из которых может формироваться на одном из предыдущих шагов решения.

Простейший язык описания задачи обработки изображений может просто использовать имена операций, реализующих соответствующий шаг решения, и идентификаторы элементов решения.

Например, задача выделения контуров на изображении может быть записана следующим образом

$$\mathbf{b} = Op\_Sobel(\mathbf{a}),\ \mathbf{g} = Op\_Thresholding(\mathbf{b},128),$$

где $\mathbf{a}$ –исходное изображение, $\mathbf{b}$ – промежуточное изображение, являющееся результатом применения фильтра Собела (операции Op_Sobel) к исходному изображению, $\mathbf{g}$ – результирующее изображение, ре-

зультат применения пороговой обработки (операции $Op\_Thresholding$) к промежуточному изображению.

При переходе к формальному описанию данной задачи нет необходимости в детальной конкретизации отдельных шагов технологии обработки, это задача разработчика прикладной программы, реализующей определенный шаг решения. В данном случае существенен только информационный тип программы обработки изображения.

Формальная запись алгоритма задачи выделения контуров на изображении может иметь следующий вид

$$\mathbf{b} = LNO(\mathbf{a}), \mathbf{g} = PO(\mathbf{b}),$$

откуда можно видеть, что исходное изображение $\mathbf{a}$ подвергается преобразованию с помощью операции локальной обработки скользящим окном, а затем полученное изображение $\mathbf{b}$ обрабатывается с применением операции поэлементного преобразования. Тот факт, что при пороговой обработке использовался порог со значением 128, совершенно несущественен при формальном описании данной задачи.

При формальном описании алгоритмов обработки изображений в терминах типовых операций обработки широко используют различные варианты графовых моделей [5], например Image Application Task Graph (IATG) [6].

Шаги решения с соответствующими элементами решения представляются в виде графа решения данной задачи. Множеству шагов решения (операциям Φ^I алгоритма) ставится во взаимно-однозначное соответствие некоторое множество точек – вершин. Если элемент решения одной операции используется при выполнении другой операции, то соответствующие вершины соединяются дугой, направленной из той точки, где формируется данный элемент решения. Из каждой вершины выходит столько дуг, сколько формируется на данном шаге изображений и параметров. В случае, когда элемент решения является аргументом для нескольких операций, выходящих дуг может быть больше.

Расширения графовых моделей, связанные с учетом временных характеристик, формируются путем придания некоторых весовых функций как вершинам, так и дугам графа решения.

Примером графовой модели описания информационной структуры задачи обработки изображений является IATG-модель [6], которая представляет собой взвешенный направленный ациклический граф $G(V, E, w, c)$, где:

- V – конечное множество вершин, представляющих шаги решения (типовые операции обработки изображений или реализованные пользователем алгоритмы);
- E – множество дуг (направленных ребер) графа, которые соответствуют условиям предшествования между операциями: $e = (u, v) \in E$, если $u \prec v$;
- w – весовая функция $w: V \to N^*$, которая задает вес (время обработки) каждой вершины (операции обработки), веса операций – целые положительные числа;
- c – функция коммуникационных затрат $c: V \to N^*$, которая задает вес (коммуникационные затраты по времени) каждой дуги, коммуникационные веса – целые положительные числа.

Вычислительные затраты операции обработки, выполняемой на одном процессоре, оцениваются по результатам тестовых экспериментов с использованием специальных средств профильного анализа. Вычислительные затраты при параллельной организации вычислений внутри некоторой операции обработки оцениваются с помощью закона Амдала

$$T_{exec}(i, p(i)) = \left(\alpha(i) + \frac{1-\alpha(i)}{p(i)} \right) \tau(i),$$

где i – номер операции (шага решения), $p(i)$ – число процессоров, на которых операция i выполняется, $\tau(i)$ – время выполнения операции на одном процессоре, $\alpha(i)$ – доля операций данного шага решения, которые не могут быть распараллелены.

Коммуникационные затраты появляются в том случае, когда задачи, реализующие отдельные шаги решения, размещаются на разных компьютерах. Для оценки коммуникационных затрат используют модель Хокни [7], в соответствии с которой

$$T_{comm}(i, j) = t_s + L t_b,$$

где t_s, t_b – затраты на подготовку сообщения к передаче (латентность) и передачу от вершины i к вершине j одного байта сообщения соответственно, L – размер передаваемого сообщения.

Графовые модели хорошо отражают макроструктуру алгоритма и, соответственно, используются для выявления возможного параллелизма по вычислениям. Для анализа параллелизма по данным необходимо перейти на более детальный уровень абстракции модели и описать информационную структуру операций, реализующих соответствующие шаги решения. Выразительные возможности для выполнения преобразований моделей на этом уровне существенно менее наглядны.

Более адекватными с точки зрения иерархичности информационной модели представляются модели, основанные на методологии теории алгебры процессов, которые, с одной стороны, более точно отражают последовательный характер организации вычислений в программе, а с другой стороны, позволяют задать способ организации параллелизма путем явного описания взаимодействия процессов при параллельной обработке через механизмы, которые хорошо соответствуют реальным реализациям в программных системах.

Основные понятия теории взаимодействующих последовательных процессов

За основу формальной модели процессов параллельной обработки изображений возьмем математическую теорию взаимодействующих последовательных процессов Ч. Хоара – алгебру CSP [8]. Основными позитивными чертами этой теории являются:

- развитое понятие процесса, учитывающее особенности синхронизации и параллельного выполнения процессов, включая модель обмена данными между различными процессами;
- удобная техника работы с протоколами (последовательностью событий) процессов, позволяющая легко выделять и анализировать результаты работы модели;
- возможность эквивалентного отображения теории на языки высокого уровня.

В основе метода формального описания взаимодействующих последовательных процессов лежит математическая теория, описанная в виде систематического набора алгебраических законов.

Область применения теории CSP – спецификация, разработка и реализация вычислительных систем, которые непрерывно действуют и взаимодействуют со своим окружением. Основная идея заключается в том, что эти системы можно разложить на параллельно работающие подсистемы, взаимодействующие как друг с другом, так и со своим системным окружением. Основным объектом рассмотрения является процесс, описываемый как последовательность ограниченного набора событий, выбранных для его описания. Данное множество событий называется алфавитом процесса. Алфавит считается постоянным, заранее определенным свойством объекта. При выборе алфавита проводят некоторое упрощение: не рассматриваются многие действия и свойства, которые не влияют на исследуемые моделью характеристики.

На уровне макроопераций описание задачи обработки изображений может строиться как композиция процессов, реализующих шаги решения данной задачи. Например, описание задачи выделения контуров на изображении может быть задано в виде последовательной композиции двух процессов

$$CONTOUR(\mathbf{a},\mathbf{g}) = LNO(\mathbf{a},\mathbf{b}); PO(\mathbf{b},\mathbf{g}),$$

где $LNO(\mathbf{a},\mathbf{b})$ – процесс, выполняющий преобразование исходного изображения $\mathbf{a}$ в изображение $\mathbf{b}$ с использованием операции локальной обработки скользящим окном, $PO(\mathbf{b},\mathbf{g})$ – процесс, выполняющий поэлементное преобразование изображения $\mathbf{b}$ и формирующий при этом изображение $\mathbf{g}$, операция « ; » обозначает последовательную композицию этих процессов. Данная модель соответствует обычной последовательной программе, выполняющейся на одном процессоре. Для описания систем с параллельным выполнением процессов используется операция *параллельной композиции.* Например, модель параллельной программы, реализующей задачу выделения контуров на изображении, на уровне макроопераций записывается в следующем виде

$$CONTOUR(\mathbf{a},\mathbf{g}) = LNO(\mathbf{a},\mathbf{b}) \,||\, PO(\mathbf{b},\mathbf{g}).$$

Как именно осуществляется распараллеливание вычислений в такой параллельной программе определяется при описании процессов, из которых составлена данная модель.

Для описания поведения объектов (процессов) в [8] определяется следующая система обозначений. Пусть x - событие, а P - процесс. Тогда

$$(x \to P)$$

описывает процесс, который вначале участвует в событии x, а затем ведет себя в точности как P. Алфавит процесса P обозначается αP. Процесс $(x \to P)$ имеет по определению тот же алфавит, что и P, поэтому это обозначение можно использовать только при условии, что x принадлежит тому же алфавиту. Более формально:

$$\alpha(x \to P) = \alpha P, \quad if \ x \in \alpha P.$$

Процесс с алфавитом A, в котором не происходит ни одно событие из A, называется $STOP_A$. Этот процесс чаще всего возникает в результате ошибок проектирования или использования (дедлок). Однако процесс может прекратить работу, если он выполнил все, что ему полагалось. О таком процессе говорят, что он успешно завершился. Завершение рассматривают как специальное событие, обозначаемое символом $\surd$ («успех»), чтобы отличать его от процесса $STOP$. Процесс, имеющий в алфавите символ $\surd$, называется последовательным. Определяют специальный процесс $SKIP_A$, который ничего не делает, но благополучно завершается: $\alpha SKIP_A = A \cup \{\surd\}$.

Используемую выше префиксную запись можно использовать для полного описания поведения последовательного процесса, который получает два операнда, выполняет операцию сложения, выводит результат и останавливается. Например:

$$SUM = (op1 \to (op2 \to (add \to (out \to SKIP)))).$$

Скобки в записи можно отбросить, считая операцию $\to$ ассоциативной справа. В записи с операцией $\to$ справа всегда стоит процесс, а слева – отдельное событие. Описание процесса, начинающееся с префикса, называется предваренным. Если $F(X)$ – предваренное выражение, содержащее имя процесса X, а множество событий A – алфавит X, то утверждается, что уравнение $X = F(X)$ имеет единственное решение в алфавите A, которое обозначается выражением $\mu X : A.F(X)$. Здесь X является локальным именем (связанной переменной) и может произвольно изме-

няться, поскольку $\mu X : A.F(X) = \mu Y : A.F(Y)$. В записи с помощью μ-оператора обычно опускается явное упоминание алфавита A, если это понятно из контекста или содержания процесса.

Такое рекурсивное задание процесса легко обобщается на случай решения систем уравнений более чем с одним неизвестным. Для достижения результата необходимо, чтобы правые части всех уравнений были предваренными, а каждый неизвестный процесс входил ровно один раз в правую часть одного из уравнений. Использование переменных с индексами позволяет описывать бесконечные системные уравнения.

Определим более формально упоминаемое ранее понятие последовательной композиции: если *P* и *Q* – последовательные процессы с одним и тем же алфавитом, то их последовательная композиция (*P*; *Q*) представляет собой процесс, ведущий себя сначала как *P*, а после успешного завершения *P* продолжающий вести себя как *Q*. Если успешного завершения *P* не происходит, то не завершается и (*P*; *Q*). Используя последовательную композицию, можно определить бесконечный цикл как особый случай рекурсии:

$$*P = \mu X.(P; X) = P; P; P; \ldots .$$

Операция параллельной композиции: если *P* и *Q* – процессы, для которых, вообще говоря, $\alpha P \neq \alpha Q$, то их параллельная композиция обозначается через $(P \| Q)$ и является процессом, алфавит которого $\alpha(P \| Q) = \alpha P \cup \alpha Q$, который ведет себя как система, составленная из процессов *P* и *Q*, причем события из алфавита $\alpha P \cap \alpha Q$ пошагово синхронизированы и требуют одновременного участия *P* и *Q*, события из алфавита $\alpha P \setminus \alpha Q$ не имеют никакого отношения к *Q*, который не способен ни контролировать, ни даже замечать их, аналогично *Q* самостоятельно участвует в событиях из алфавита $\alpha Q \setminus \alpha P$.

Определим также важные аспекты последовательного программирования: присваивания, условные операторы и циклы.

Если *x* – программная переменная, *e* – выражение, а *P* – процесс, то $(x := e; P)$ – это процесс, ведущий себя как *P*, но только начальное значение *x* задается равным значению выражения *e*. Более формально:

$$(x := e) = (x := e; SKIP).$$

Далее приведем некоторые законы присваивания необходимые в дальнейшем.

В законах для присваивания *x* и *y* обозначают списки различных переменных; *e*, $f(x)$, $f(e)$ обозначают списки выражений, возможно, содержащих вхождения переменных из *x* или *y*; и если $f(x)$ содержит x_i, то $f(e)$ содержит e_i для всех индексов *i*. Для простоты считается, что все выражения в законах имеют результат для любых значений содержащихся в них переменных.

$$(x := x) = SKIP,$$

$$(x := e; x := f(x)) = (x := f(e)).$$

Для эффективного использования присваивания в параллельных процессах необходимо наложить ограничение: ни одна переменная из тех, которым было присвоено значение внутри одного процесса, не может использоваться в другом. Чтобы соблюсти это ограничение, в алфавит последовательного процесса вводятся две категории символов: $var(P)$ – множество переменных, которым можно присваивать значения внутри *P*, $acc(P)$ – множество переменных, доступных в выражениях внутри *P*.

Все переменные, которые можно изменять, являются также и доступными:

$$var(P) \subseteq acc(P) \subseteq \alpha P .$$

Определим по аналогии $acc(e)$ как множество переменных, встечающихся в *e*. Если *P* и *Q* объединяются оператором параллельной композиции, то

$$var(P) \cap acc(Q) = var(Q) \cap acc(P) = \{\ \}.$$

Пусть выражение *b* вычисляет истинность логической функции (значение его либо истина, либо ложь). Если *P* и *Q* – процессы, то

$$P\ <\ b\ >\ Q \quad (P \textbf{ if } b \textbf{ else } Q) -$$

это процесс, ведущий себя как *P*, если начальное значение *b* истинно, или как *Q*, если начальное значение *b* ложно.

Традиционный цикл

while b do Q

записывается как

$$b\ *\ Q.$$

Рекурсивно его можно определить как

$$b*Q = \mu X.((Q; X) < b > SKIP).$$

Для упрощения формирования моделей технологий обработки изображений путем композиции типовых операций обработки введем следующее соглашение. Так как один и тот же процесс может встречаться в сети, описывающей методику обработки, многократно, но при этом работать с различными изображениями, то, чтобы не переопределять процесс, меняя обозначения событий, введем возможность переименования событий, т.е. своего рода формальные параметры в описании процесса. Если процесс определен следующим образом:

$$P(x, y) = a \to x \to y \to SKIP,$$

то параллельная композиция $P2 = P(b, c) \| P(c, d)$ фактически означает, что

$$P2 = (a \to b \to c \to SKIP)$$
$$\| (a \to c \to d \to SKIP).$$

В перечень формальных параметров целесообразно вносить имена каналов, по которым процесс осуществляет взаимодействие, чтобы определять связи процессов не при их описании, а при записи их композиции. В формальные параметры можно вынести и событие (или функцию), определяющее собственно обработку, производимую процессом, тем самым мы получаем описание некоторого класса процессов, выполняющих различную обработку сходным образом.

Модель представления цифровых изображений

Перед тем как перейти к описанию моделей основных типовых операций обработки изображений, которые будут выступать как алфавит на уровне метаописания задачи, определим основные понятия, связанные с моделью объекта обработки – изображения.

Хорошей основой для описания изображений как структур данных служит алгебра изображений Риттера [9]. Достоинством данной модели описания изображений является явное разделение структур данных на структуру пространственной организации (взаимозависимости) пикселов изображения – области определения и структуру области значений изображения, т.е. структуру, определяющую типологию изображения, начиная от бинарного и кончая многоканальными, с использованием различных цветовых моделей. Таким образом, можно выделить аспекты реализации операций обработки изображений, связанные с пространственной организацией данных изображений.

Цифровое изображение представляется в вычислительных системах в виде множества пикселов. С каждым пикселом сопоставляются его координаты (точка с координатами) и значение. Значение пиксела изображения $\mathbf{a}$ с координатами $\mathbf{x}$ обозначается через $\mathbf{a}(\mathbf{x})$.

Множество всех координат изображения является его областью определения и обозначается прописными буквами жирного написания, например, $\mathbf{X}$, $\mathbf{Y}$ или $\mathbf{Z}$. В общем случае область определения изображения является топологией точечного множества из дискретного n-мерного пространства $\mathbb{Z}^n$. Но, как правило, и в данной работе, в частности, рассматриваются двумерные изображения, т.е. $\mathbf{X} \subset \mathbb{Z}^2$. Множество точек ограничено по каждой координате, образуя прямоугольник.

Таким образом, для двумерного изображения

$$\mathbf{X} = \{(x_1, x_2) \in \mathbb{Z}^2 : o_1 \le x_1 \le o_1 + n_1 - 1,$$
$$o_2 \le x_2 \le o_2 + n_2 - 1\},$$

где $\mathbf{o} = (o_1, o_2)$ – начало координат изображения, n_i – размер области определения по i-ой координате. Фиксируя начало координат в нуле и учитывая обозначения $\mathbb{Z}_n = \{0,1,...,n-1\}$ или $\mathbb{Z}_n^+ = \{1,2,...,n\}$, получаем

$$\mathbf{X} = \{(x_1, x_2) \in \mathbb{Z}^2 : x_1 \in \mathbb{Z}_{n_1}, x_2 \in \mathbb{Z}_{n_2}\}.$$

Множество всех значений пикселов изображения $\mathbf{a}(\mathbf{x})$ является областью значений изображения и обозначается как $\mathbb{F}$. В качестве значения пиксела может выступать вектор скалярных значений. Скалярное значение может иметь любой из типов данных начиная с бинарного и кончая действительным. Множество всех изображений, имеющих область значений $\mathbb{F}$ и область определения $\mathbf{X}$, обозначается как $\mathbb{F}^{\mathbf{X}}$.

Суммируя введенные обозначения, изображение $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ (т.е. $\mathbf{a} : \mathbf{X} \to \mathbb{F}$) является сокращенной записью следующей нотации

$$\{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{x} \in \mathbf{X} \subset \mathbb{Z}^2, \mathbf{a}(\mathbf{x}) \in \mathbb{F} \subset \{\mathbb{Z}^m, \mathbb{R}^m, \mathbb{C}\}\}.$$

Множество $\mathbb{F}$ определяет тип изображения, например, при $\mathbb{F} \in \{0,1\}$ изображение является бинарным, черно-белое полутоновое изображение имеет область значений в виде множества рациональных чисел $\mathbb{F} \in \mathbb{R}$, у цветного полутонового изображения значения яркости представляются вектором в пространстве рациональных чисел $\mathbb{F} \in \mathbb{R}^3$.

Полезно выделить следующие элементы пространственного описания изображений:

- множество точек, принадлежащих одной строке,
- множество точек, принадлежащих одному столбцу,
- множество точек, принадлежащих локальной окрестности некоторой точки.

Множество точек, принадлежащих одной строке, формируется путем фиксации значения координаты x_1 точек $\mathbf{x} \in \mathbf{X}$ и обозначается

$$\mathbf{X}_{r_i} = \mathbf{X}\big|_{x_1 = i} = \{(x_1, x_2) : x_1 = i, x_2 \in \mathbb{Z}_{n_2}\}. \quad (1)$$

Аналогично множество точек, принадлежащих одному столбцу, формируется путем фиксации значения координаты x_2 точек $\mathbf{x} \in \mathbf{X}$ и обозначается

$$\mathbf{X}_{c_j} = \mathbf{X}\big|_{x_2 = j} = \{(x_1, x_2) : x_1 \in \mathbb{Z}_{n_1}, x_2 = j\}.$$

Произвольная прямоугольная окрестность вокруг точки $\mathbf{x}$, симметрично расположенная относительно её, используется при локальной обработке изображений.

$$Q_{k,l}(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} = (x_1 \pm i, x_2 \pm j), i \in \mathbb{Z}^+_k, j \in \mathbb{Z}^+_l\}.$$

Размер данного прямоугольного окна равен $(2k+1) \times (2l+1)$.

При параллельной обработке данные изображений распределяются по отдельным узлам распределенной вычислительной системы. Структуры дан-

ных одного изображения, размещенные по разным узлам, образуют единую логическую сущность, называемую распределенным изображением.

Часть данных распределенного изображения, хранящаяся на отдельном узле, называется частичным изображением. Дополнительно с каждым частичным изображением хранится информация, описывающая параметры разбиения и размещения. Эта информация содержит (но не ограничивается) описание и параметры топологии распределенной вычислительной системы, начало координат и размеры области определения исходного и частичного изображения, тип используемой декомпозиции. Частичное изображение, являющееся частью изображения *a* и размещенное на вычислительном узле *i*, обозначим через $\mathbf{a}_{p_i}$, его область определения – $\mathbf{X}_{p_i}$. Для любого изображения $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, распределенного по *n* узлам, справедливо соотношение

$$\bigcup_{i=0}^{n-1} \mathbf{X}_{p_i} = \mathbf{X}.$$

Совокупность всех *n* частичных изображений, соответствующих изображению *a*, образует единую логическую структуру данных – распределенное изображение, обозначаемое как $\mathbf{a}_d$. Область определения распределенного изображения $\mathbf{X}_d$ задается через дизъюнктивное объединение областей определения всех его частичных изображений. Области определения частичных изображений, которые формируют распределенное изображение, могут быть образованы с использованием различных способов декомпозиции.

Типология декомпозиции области определения распределенного изображения включает выбор размерности разбиения и способа взаимного пересечения (сопряжения) отдельных частей изображения. Возможные варианты декомпозиции изображения сведены в таблицу 1.

Таблица 1. Обозначения способов декомпозиции изображений

Вид декомпозиции	Способ сопряжения фрагментов	
	Без пересечения	С частичным перекрытием фрагментов на *k* строк (и *l* столбцов)
Одномерная	$\mathbf{a}_d = \coprod_{i=1,P} \mathbf{a}_{p_i}$ или $\mathbf{a}_d = \coprod_{P} \mathbf{a}_{p_i}$	$\mathbf{a}_d = \coprod_{i=1,P}^{k} \mathbf{a}_{p_i}^{\pm k}$ или $\mathbf{a}_d = \coprod_{P}^{k} \mathbf{a}^{\pm k}{}_{p_i}$
Двумерная	$\mathbf{a}_d = \coprod_{i=1,P;\, j=1,P} \mathbf{a}_{p_{i,j}}$ или $\mathbf{a}_d = \coprod_{P^2} \mathbf{a}_{p_{i,j}}$	$\mathbf{a}_d = \coprod_{i=1,P;\, j=1,P}^{k,l} \mathbf{a}_{p_{i,j}}^{\pm k,l}$ или $\mathbf{a}_d = \coprod_{P^2}^{k,l} \mathbf{a}_{p_{i,j}}^{\pm k,l}$

Отдельно можно указать вариант репликации данных, т.е. при формировании распределенного изображения частичные изображения полностью дублируют исходное изображение, что обозначается как

$$\mathbf{a}_d = \coprod_{P}^{*} \mathbf{a}^{*}{}_{p_i}.$$

Рассмотренная выше растровая модель представления цифрового изображения описывает логический, или абстрактный, уровень представления.

Логической уровень представления изображений в виде двумерной матрицы отсчетов используется при описании алгоритмов обработки.

Существует еще и физический, или конкретный, уровень представления данных изображений, который описывает то, как физически хранятся данные в компьютерах.

Физической структурой данных в вычислительных системах обычно является одномерный массив, поскольку подавляющее число современных компьютеров имеет запоминающие устройства с линейным адресным доступом. Таким образом, физическое представление данных не совпадает с логическим, и идентификатором пиксела (элемента изображения) после перенесения его в память служит уже не его положение (координаты) в *n*-мерном массиве, а адрес ячейки памяти, где хранится значение этого элемента изображения. Чтобы не хранить дополнительно к каждому элементу его координаты в *n*-мерном изображении, должен быть известен способ сканирования (развертки) массива при размещении его в памяти, в этом случае по адресу хранения (индексу в одномерном массиве) можно однозначно восстановить его координаты и наоборот. Это является неявной схемой привязки хранимых элементов данных к их координатам на изображении. Таким способом представляются в компьютере все растровые изображения.

Пусть $t \in \mathbf{T} \subset \mathbb{Z}^1$ – адрес хранения, или индекс, в одномерном массиве, где располагается пиксел **a**(**x**). Соответствующее множество индексов пикселов, описывающее физический уровень представления, задается соотношением

$$\mathbf{T} = \left\{ t \in \mathbb{Z}^1 : 0 \le t \le n_1 n_2 - 1 \right\}.$$

Преобразование развертки $t = \tau(\mathbf{x})$, где $\tau : \mathbf{X} \to \mathbf{T}$ определяет способ размещения (сканирования) элементов многомерного массива в линейно-упорядоченной памяти (в виде одномерного массива).

Например, широко известная телевизионная развертка для двумерного изображения определяется с помощью следующего соотношения $t = x_1 n_2 + x_2$.

Большинство алгоритмов использует логический уровень представления изображений. В процессе реализации алгоритмов программа осуществляет согласование логического и физического представлений, происходит уточнение информационной структуры алгоритма путем явного указания последовательности обработки.

Модели основных операций обработки изображений

Основываясь на обозначениях теории взаимодействующих последовательных процессов, определим модели основных операций обработки изображений.

Формальная модель операции редукции изображения

Пусть есть изображение $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ и определена некоторая операция γ, коммутативная и ассоциативная, которая формирует некоторую первичную характеристику изображения

$$\prod \mathbf{a} = \mathbf{a}(\mathbf{x}_1)\gamma\mathbf{a}(\mathbf{x}_2)\gamma\cdots\gamma\mathbf{a}(\mathbf{x}_n), \text{ где } n = n_1 \cdot n_2,$$

например, сумму всех значений пикселов изображения

$$\sum \mathbf{a} = \sum_{\forall \mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x}_1) + \mathbf{a}(\mathbf{x}_2) + \cdots + \mathbf{a}(\mathbf{x}_n).$$

Выражение $\forall \mathbf{x} \in \mathbf{X}$ определяет только то, что нужно перебирать все элементы множества $\mathbf{X}$, но не указывает, в каком порядке это выполнять. Между тем, для формирования информационной модели алгоритма необходимо определить порядок выборки данных, порядок выполнения операций.

Для программ обработки изображений порядок перебора элементов изображения традиционно определяется с помощью так называемой «телевизионной» развертки, слева направо по строке и сверху вниз по строкам. Т.е. если изображение в программе определено в соответствии с логическим уровнем представления в виде двумерного массива, то выполнение операции редукции (в частности, суммирования) можно реализовать в виде следующего примерного кода на языке программирования C.

```
int a[k][l];
int s=0;
for(int i=0; i < k; i++)
    for(int j=0; j < l; j++) s += a[i][j];
```

В более общем виде выражение в теле цикла может быть следующим

```
for(int i=0; i < k; i++)
    for(int j=0; j < l; j++)
              s += f(i, j, a[i][j], p1,…,pN);
```

где `f(i, j, a[i][j], p1,…,pN)` – некоторая функция, в общем случае зависящая от координат обрабатываемого пиксела и набора параметров `p1,…,pN`.

При использовании физического уровня представления изображения в программе при выполнении итераций используется естественный порядок размещения элементов (пикселов) изображения в памяти компьютера в виде одномерного массива и код реализации операции рекурсии общего вида может иметь следующий вид

```
int a[n];
int s=0;
for(int i=0; i < n; i++)
    s += f(x1(i), x2(i), a[i], p1,…,pN);
```

Здесь дополнительно используются `x1(i)`, `x2(i)` – функции вычисления двумерных координат пиксела по его индексу в массиве хранения данных изображения.

Формальная модель этого процесса в терминах алгебры CSP в простейшем виде может быть записана в виде определения следующего процесса

$$RO = \left(s := 0; i := 0; (i < n) * \left(s := s + a_i; i := i + 1\right)\right),$$

здесь a_i – значения яркости пиксела изображения $\mathbf{a}(\mathbf{x}_i) \in \mathbb{F}^{\mathbf{X}}$ с координатами $\mathbf{x}_i$.

Более общее описание имеет следующий вид

$$RO(\mathbf{a}, s, f, \gamma) =$$
$$= \left(s := 0; i := 0; (i < n) * \left(s := s\gamma f(a_i); i := i + 1\right)\right)$$

и означает применение ко всем скорректированным с помощью преобразования f пикселам изображения $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ некоторой операции редукции γ, результат которой формируется в переменной s.

Если исходное изображение $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ преобразовано посредством некоторой произвольной декомпозиции в распределенное изображение

$$\mathbf{a}_d = \coprod_P \mathbf{a}_{p_i},$$

то операция редукции может быть выполнена параллельно над частичными изображениями $\mathbf{a}_{p_i} \in \mathbb{F}^{\mathbf{X}}$ с последующим объединением частичных результатов s_{p_i}.

$$RO(\mathbf{a}_d, s, f, \gamma) =$$
$$= \left(RO(\mathbf{a}_{p_1}, s_{p_1}, f, \gamma) \| \cdots \| RO(\mathbf{a}_{p_P}, s_{p_P}, f, \gamma)\right);$$
$$\left(s := s_{p_1}\gamma\cdots\gamma s_{p_P}\right)$$

или в эквивалентной записи

$$RO(\mathbf{a}_d, s, f, \gamma) = \left(\|_P RO(\mathbf{a}_{p_i}, s_{p_i}, f, \gamma)\right); \left(s := \prod_P s_{p_i}\right).$$

Из последней записи легко видеть, что операция редукции является глобальной, и, несмотря на возможность распараллеливания большей части вычислений, на заключительной стадии все равно необходимо выполнить коллективную операцию редукции частичных результатов, полученных в параллельных ветках.

Формальная модель операции поэлементного преобразования

Рассмотрим некоторую произвольную унарную операцию поэлементного преобразования. В процессе обработки значение каждой точки исходного изображения $\mathbf{a}\in\mathbb{F}^{\mathbf{X}}$ преобразуется в значение соответствующей ей точки выходного изображения $\mathbf{c}\in\mathbb{F}^{\mathbf{X}}$:

$$\mathbf{c}(\mathbf{x})=F\left(\mathbf{a}(\mathbf{x})\right),\ \forall\mathbf{x}\in\mathbf{X}.$$

Здесь также выражение $\forall\mathbf{x}\in\mathbf{X}$ определяет, что нужно перебирать все элементы множества $\mathbf{X}$, причем порядок выборки данных, порядок выполнения операций не существенен.

Неявно предполагается, что тип развертки (т.е. порядок перебора пикселов) исходного изображения уже определен тем, как формировались данные на предыдущих этапах обработки. Если операция выполняется в отдельной программе, то тип развертки исходного изображения определяется тем форматом, который используется при хранении данных на ВЗУ.

Тип развертки результирующего изображения целесообразно выбрать так, чтобы он совпадал с разверткой исходного ихображения.

При использовании физического уровня представления изображений в программе при выполнении итераций используется естественный порядок размещения элементов (пикселов) изображения в памяти компьютера в виде одномерного массива и код реализации унарной операции поэлементного преобразования общего вида может иметь следующий вид

```
int a[n], c[n];
for(int i=0; i < n; i++)
    c[i] = f(x1(i), x2(i), a[i], p1,…,pN);
```

где `x1(i), x2(i)` – функции вычисления двумерных координат пиксела по его индексу в массиве хранения данных изображения; `p1,…,pN` – набор параметров поэлементного преобразования `f`.

Здесь предполагается, что области определения изображений **a** и **c** заданы одинаковым образом. В результате выполнения операции в оперативной памяти будет сформирован массив, который будет содержать данные результирующего изображения, тип развертки которого будет совпадать с разверткой массива, хранящего данные исходного изображения.

Фактически в данном коде имеются две структуры данных, которые итерируются синхронно, т.е. с использованием одного указателя.

Формальная модель этой операции в терминах алгебры CSP записывается в виде следующего процесса

$$PO(\mathbf{a},\mathbf{c},f)=\left(i:=0;(i<n)*\left(c_i:=f(a_i);i:=i+1\right)\right),$$

где предполагается, что ранее в системе уже определены значения пикселов изображения $\mathbf{a}\in\mathbb{F}^{\mathbf{X}}$, они обозначаются как $\{a_i\}$, в процессе обработки формируются значения пикселов изображения $\mathbf{c}\in\mathbb{F}^{\mathbf{X}}$, которые обозначаются как $\{c_i\}$. Более формально:

$$acc\left(PO(\mathbf{a},\mathbf{c},f)\right)=\{a_i\}=\mathbf{a},$$

$$var\left(PO(\mathbf{a},\mathbf{c},f)\right)=\{c_i\}=\mathbf{c}.$$

Произвольная бинарная операция поэлементной обработки формирует значение каждой точки выходного изображения $\mathbf{c}\in\mathbb{F}^{\mathbf{X}}$ как результат некой операции над соответствующими точками двух исходных изображений $\mathbf{a}\in\mathbb{F}^{\mathbf{X}}$ и $\mathbf{b}\in\mathbb{F}^{\mathbf{X}}$:

$$\mathbf{c}(\mathbf{x})=F\left(\mathbf{a}(\mathbf{x}),\mathbf{b}(\mathbf{x})\right),\ \forall\mathbf{x}\in\mathbf{X}.$$

Формальная модель этой операции в терминах алгебры CSP описывает процесс синхронной итерации с использованием одного указателя уже трех структур данных $\{\mathbf{a}_i\}$, $\{\mathbf{b}_i\}$ и $\{\mathbf{c}_i\}$

$$PO2(\mathbf{a},\mathbf{b},\mathbf{c},f)=$$
$$=\left(i:=0;(i<n)*\left(c_i:=f(a_i,b_i);i:=i+1\right)\right),$$

причем

$$acc\left(PO2(\mathbf{a},\mathbf{b},\mathbf{c},f)\right)=\{a_i\}\cup\{b_i\}=\mathbf{a}\cup\mathbf{b},$$

$$var\left(PO2(\mathbf{a},\mathbf{b},\mathbf{c},f)\right)=\{c_i\}=\mathbf{c}.$$

При распараллеливании поэлементных операций параметры и тип декомпозиции всех итерируемых структур данных должны совпадать. Какая из структур является при этом определяющей?

Учитывая необходимость обеспечения сбалансированности вычислительной нагрузки процессоров при параллельной обработке, а это проще выполнять перераспределяя вычисления, связанные с формированием значения пиксела результирующего изображения, в качестве основной структуры, которая определяет параметры и тип декомпозиции всех итерируемых структур данных, целесообразно выбрать результирующее изображение. При этом, учитывая ограничения на множество присваиваемых переменных параллельных процессов, можно показать, что декомпозиция результирующего изображения должна выполняться на непересекающиеся части.

Таким образом, при распараллеливании операций поэлементной обработки необходимо выполнить декомпозицию результирующего изображения на непересекающиеся фрагменты. Размер частичных изображений определяется исходя из априорной информации о соотношении производительности используемых при обработке вычислительных систем. В случае гомогенных параллельных вычислительных систем используется простейшая декомпозиция на одинаковые частичные изображения

$$\mathbf{c}_d=\coprod_P\mathbf{c}_{p_i},$$

формирующая соответствующий набор частичных множеств определения

$$\mathbf{X}_d = \coprod_P \mathbf{X}_{p_i},$$

на основании которого, в свою очередь, будет определена декомпозиция исходных изображений

$$\mathbf{a}_d = \coprod_P \mathbf{a}_{p_i},\ \mathbf{b}_d = \coprod_P \mathbf{b}_{p_i}.$$

Процессы, описывающие параллельную поэлементную операцию над изображениями, имеют следующий вид

$$PO(\mathbf{a}_d, \mathbf{c}_d, f) =$$
$$= \left(PO\left(\mathbf{a}_{p_1}, \mathbf{c}_{p_1}, f\right) \| \cdots \| PO\left(\mathbf{a}_{p_P}, \mathbf{c}_{p_P}, f\right)\right),$$
$$PO2(\mathbf{a}_d, \mathbf{b}_d, \mathbf{c}_d, f) =$$
$$= \left(PO2\left(\mathbf{a}_{p_1}, \mathbf{b}_{p_1}, \mathbf{c}_{p_1}, f\right) \| \cdots \| PO2\left(\mathbf{a}_{p_P}, \mathbf{b}_{p_P}, \mathbf{c}_{p_P}, f\right)\right),$$

или в эквивалентной записи

$$PO(\mathbf{a}_d, \mathbf{c}_d, f) = \|_P PO\left(\mathbf{a}_{p_i}, \mathbf{c}_{p_i}, f\right),$$

$$PO2(\mathbf{a}_d, \mathbf{b}_d, \mathbf{c}_d, f) = \|_P PO2\left(\mathbf{a}_{p_i}, \mathbf{b}_{p_i}, \mathbf{c}_{p_i}, f\right).$$

Операция поэлементного преобразования изображений допускает полное распараллеливание на основе непересекающейся декомпозиции.

Формальная модель операции локальной обработки скользящим окном

Рассмотрим типовую операцию локальной обработки скользящим окном, которая формирует изображение $\mathbf{c} \in \mathbb{F}^{\mathbf{X}}$, значение в каждой точке $\mathbf{x} \in \mathbf{X}$ которого является результатом некоторого преобразования над прямоугольной, симметрично расположенной относительно текущей точки $\mathbf{x} \in \mathbf{X}$ окрестностью $Q_{k,l}(\mathbf{x})$ точек исходного изображения $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$

$$\mathbf{c}(\mathbf{x}) = F\left(\left\{\mathbf{a}(\mathbf{y}) : \mathbf{y} \in Q_{k,l}(\mathbf{x})\right\}\right),\ \forall \mathbf{x} \in \mathbf{X}.$$

При выполнении данной операции явно задается только итерация по точкам результирующего изображения. Итерация по точкам исходного множества осуществляется путем косвенной итерации по точкам окрестности текущей точки, соответствующей каждому положению вычисляемой точки результирующего изображения. В процессе косвенной итерации происходит перебор значений смещений индексов, которые необходимо добавить к текущему индексу первичной итерации. При вычислениях, производимых в процессе выполнения косвенной итерации, часто используется массив коэффициентов шаблона преобразования, на соответствующие значения которого умножается значение точки исходного изображения.

Формальную модель операции локальной обработки скользящим окном удобно записать в виде построчной итерации, т.е. процесс обработки заключается в последовательном формировании строк результирующего изображения

$$LNO(\mathbf{a}, \mathbf{c}, f) =$$
$$= \left(i := 0; (i < n_1) * \left(c_{r_i} := f\left(\bigcup_{j=i-k}^{k} a_{r_{i+j}}\right); i := i+1\right)\right).$$

Для строки результирующего изображения $\mathbf{c}_{r_i}$ с областью определения (1) соответствующая область определения фрагмента исходного изображения, значения отсчетов которого используются при формировании этой строки, определяется следующим образом

$$\mathbf{X}_{r_{i\pm k}} = \bigcup_{j=-k}^{k} \mathbf{X}_{r_{i+j}}.$$

Аналогично поэлементной обработке при распараллеливании операций локальной обработки скользящим окном первичным является выбор декомпозиции результирующего изображения на непересекающиеся фрагменты. Соответствующая декомпозиция исходного изображения определяется в виде одномерной декомпозиции с перекрытием на k строк

$$\mathbf{a}_d = \coprod_{i=\overline{1,P}}^{k} \mathbf{a}_{p_i}^{\pm k}.$$

Процесс, описывающий параллельный вариант локальной обработки скользящим окном над изображением, имеет следующий вид

$$LNO(\mathbf{a}_d, \mathbf{c}_d, f) = \|_P LNO\left(\mathbf{a}_{p_i}^{\pm k}, \mathbf{c}_{p_i}, f\right).$$

Операция локальной обработки изображений скользящим окном допускает полное распараллеливание на основе декомпозиции с перекрытием, размер которого определяется параметрами окна обработки.

Пример формирования параллельной модели типовой задачи обработки изображений

Рассмотрим технологию повышения контрастности изображения. Пусть реализующая ее программа включает следующие этапы: сбор статистики, в частности, математического ожидания и дисперсии значений яркости, с последующим выполнением поэлементного преобразования.

В достаточно общем виде данную технологию можно записать в виде последовательной композиции четырех процессов

$$RO(\mathbf{a}, sm, 1, +); RO(\mathbf{a}, sd, \wedge 2, +);$$
$$(m := sm / n; d := sd / n);$$
$$PO(\mathbf{a}, \mathbf{c}, f(m, d)),$$

где, в свою очередь, можно выделить процессы вычисления суммы значений яркости отсчетов изображения $RO(\mathbf{a}, sm, 1, +)$ и вычисления суммы квадратов значений яркости отсчетов изображения $RO(\mathbf{a}, sd, \wedge 2, +)$, процесс вычисления значений мате-

матического ожидания и дисперсии ($m := sm / n$; $d := sd / m$), процесс поэлементного преобразования $PO(\mathbf{a}, \mathbf{c}, f(m,d))$.

Три из составляющих эту последовательную композицию процессов могут быть преобразованы в процессы, выполняемые параллельно, т.е. представлены в следующем виде

$$RO(\mathbf{a}_d, sm, 1, +) = \\ = RO(\mathbf{a}_{p_1}, sm_{p_1}, 1, +) \| \ldots \| RO(\mathbf{a}_{p_P}, sm_{p_P}, 1, +), \quad (2)$$

$$RO(\mathbf{a}_d, sd, \wedge 2, +) = \\ = RO(\mathbf{a}_{p_1}, sd_{p_1}, \wedge 2, +) \| \ldots \| RO(\mathbf{a}_{p_P}, sd_{p_P}, \wedge 2, +), \quad (3)$$

$$PO(\mathbf{a}_d, \mathbf{c}_d, f(m,d)) = \\ = PO(\mathbf{a}_{p_1}, \mathbf{c}_{p_1}, f(m,d)) \| \ldots \| PO(\mathbf{a}_{p_P}, \mathbf{c}_{p_P}, f(m,d)).$$

Учитывая то, что процессы (2) и (3) формируют частичные суммы значений яркости и их квадратов, процесс вычисления значений математического ожидания и дисперсии необходимо переписать в следующем виде

$$(sm := \sum_{i=1}^{P} sm_{p_i};\ sd := \sum_{i=1}^{P} sd_{p_i};$$
$$m := sm / n;\ d := sd / n).$$

Используя данные представления составляющих описание задачи процессов, можно переписать общую модель в следующем виде

$$\left(RO(\mathbf{a}_{p_1}, sm_{p_1}, 1, +) \| \ldots \| RO(\mathbf{a}_{p_P}, sm_{p_P}, 1, +)\right);$$

$$\left(RO(\mathbf{a}_{p_1}, sd_{p_1}, \wedge 2, +) \| \ldots \| RO(\mathbf{a}_{p_P}, sd_{p_P}, \wedge 2, +)\right);$$

$$\left(sm := \sum_{i=1}^{P} sm_{p_i};\ sd := \sum_{i=1}^{P} sd_{p_i};\right.$$
$$\left. m := sm / n;\ d := sd / n\right);$$

$$\left(PO(\mathbf{a}_{p_1}, \mathbf{c}_{p_1}, f(m,d)) \| \ldots \| PO(\mathbf{a}_{p_P}, \mathbf{c}_{p_P}, f(m,d))\right),$$

или в эквивалентном

$$\left(\left(RO(\mathbf{a}_{p_1}, sm_{p_1}, 1, +); RO(\mathbf{a}_{p_1}, sd_{p_1}, \wedge 2, +)\right) \| \ldots\right.$$
$$\left. \ldots \| \left(RO(\mathbf{a}_{p_P}, sm_{p_P}, 1, +); RO(\mathbf{a}_{p_P}, sd_{p_P}, \wedge 2, +)\right)\right);$$

$$\left(sm := \sum_{i=1}^{P} sm_{p_i};\ sd := \sum_{i=1}^{P} sd_{p_i};\right.$$
$$\left. m := sm / n;\ d := sd / n\right);$$

$$\left(PO(\mathbf{a}_{p_1}, \mathbf{c}_{p_1}, f(m,d)) \| \ldots \| PO(\mathbf{a}_{p_P}, \mathbf{c}_{p_P}, f(m,d))\right).$$

Данная модель показывает, что программа повышения контрастности изображения не может быть реализована в полностью распараллеленном виде, т.к. на этапе формирования глобальных статистических параметров (математического ожидания и дисперсии) необходимо выполнение коллективной операции с неявным использованием синхронизации всех процессов. С помощью данной модели легко выделить два этапа, которые можно распараллелить с использованием произвольного типа декомпозиции.

Заключение

Расширение модели информационной структуры параллельной обработки изображений, необходимое для оценки времени и других характеристик эффективности реализации сформированных вариантов организации вычислений, основано на работах [10] и [11]. Каждому процессу сопоставляется функция, определяющая время выполнения процесса в зависимости от размера обрабатываемых данных. Выбор вида функции и оценка ее параметров в различных режимах функционирования процессов на программно-аппаратных средствах целевой вычислительной системы осуществляется экспериментально с использованием программных прототипов, реализующих исследуемые операции обработки.

Предложенная модель распределенного изображения в виде дизъюнктивного объединения частичных изображений, сформированных в результате применения декомпозиции заданного типа к исходному изображению, позволяет существенно упростить алфавит моделей, описывающих параллельные алгоритмы обработки изображений, основанные на декомпозиции по данным.

Выбор алгебры CSP в качестве основы формальной модели процессов параллельной обработки изображений позволяет формировать детальные описания на различных уровнях абстракции информационной модели и задать точную информационную структуру операций, реализующих соответствующие шаги решения.

Формальная модель процессов параллельной обработки изображений на основе алгебры CSP формирует иерархическое описание информационной структуры алгоритма, которое необходимо при решении задач построения эффективного отображения технологий обработки изображений на архитектуру распределенных систем.

Благодарности

Работа выполнена при поддержке гранта Президента РФ для ведущих научных школ № НШ-7414.2010.9 и грантов Российского фонда фундаментальных исследований №№ 09-07-12147-офи_м, 09-07-92421-КЭ_а, 10-07-00553-а.

Литература

1. **Воеводин, В.** Отображение проблем вычислительной математики на архитектуру вычислительных систем // Вычислительные методы и программирование. – 2000. Т. 1. – С. 37-44.
2. **Воеводин, В.** Вычислительная математика и структура алгоритмов – М.: Изд-во МГУ, 2006.
3. Методы компьютерной обработки изображений / под ред. В.А. Сойфера. изд. 2-ое, испр. – М.: Физматлит, 2003. – 784 с.

4. **Попов, С.Б.** Концепция распределенного хранения и параллельной обработки крупноформатных изображений // Компьютерная оптика. – 2007. – Т. 31, № 4. – С. 77-85.
5. **Merigot, A.** Parallel processing for image and video processing: Issues and challenges / A. Merigot and A. Petrosino // Parallel Computing. – 2008. – Vol. 34. – P. 694-699.
6. **Nicolescu, C.** A data and task parallel image processing environment / C. Nicolescu and P. Jonker // Parallel Computing. – 2002. – Vol. 28. – P. 945-965.
7. **Lastovetsky, A.** High performance heterogeneous computing / A. Lastovetsky and J. Dongarra – Hoboken, New Jersey: John Wiley & Sons, Inc., 2009.
8. **Hoare, C.** Communicating Sequential Processes – London: Prentice-Hall International, 1985.
9. **Ritter, G.** Handbook of Computer Vision Algorithms in Image Algebra / G. Ritter and J. Wilson – BocaRaton: CRC Press Inc, 1996.
10. **Roscoe, A.W.** The Theory and Practice of Concurrency.– Prentice Hall, 1997.
11. **Schneider, S.** Concurrent and Real-Time Systems: The CSP approach – Wiley, 2000.

References

1. V. Voevodin, «Mapping of computational mathematics problems to computer architecture,» Computational Methods and Programming, vol. 1, 2000, pp. 37-44. (In Russian)
2. V. Voevodin, Computational Mathematics and algorithm structure, Moskow University Publishing House, 2006. (In Russian)
3. Computer Image Processing, Part I: Basic concepts and theory, VDM Verlag Dr. Muller, 2010.
4. Popov S.B. The concept of distributed storage and parallel processing of large-size images // Computer Optics, vol. 31, № 4, 2007. (In Russian)
5. A. Merigot and A. Petrosino, «Parallel processing for image and video processing: Issues and challenges,» *Parallel Computing*, vol. 34, 2008, pp. 694-699.
6. C. Nicolescu and P. Jonker, «A data and task parallel image processing environment,» Parallel Computing, vol. 28, 2002, pp. 945-965.
7. A. Lastovetsky and J. Dongarra, *High performance heterogeneous computing*, Hoboken, New Jersey: John Wiley & Sons, Inc., 2009.
8. C. Hoare, *Communicating Sequential Processes*, London: Prentice-Hall International, 1985.
9. G. Ritter and J. Wilson, Handbook of Computer Vision Algorithms in Image Algebra, BocaRaton: CRC Press Inc, 1996.
10. Roscoe, A.W., The Theory and Practice of Concurrency.– Prentice Hall, 1997.
11. Schneider, S., Concurrent and Real-Time Systems: The CSP approach. Wiley, 2000.

MODELING THE TASK INFORMATION STRUCTURE IN PARALLEL IMAGE PROCESSING

S.B. Popov
Image Processing Systems Institute of the RAS

Abstract

An approach to building models, that describe the information structure of parallel programs for image processing, is presented. The CSP theory is chosen as the basis of a formal model of parallel processing of images. This model generates a hierarchical description of information structure of the algorithm necessary to solve problems of constructing an effective mapping of image processing tasks on the architecture of distributed systems.

Key words: information structure of the algorithm, CSP theory, parallel image processing, digital image model, image partitioning.

Сведения об авторе

Попов Сергей Борисович, кандидат технических наук, доцент Самарского государственного аэрокосмического университета имени академика С.П. Королева; старший научный сотрудник Учреждения Российской академии наук Институт систем обработки изображений РАН. E-mail*: spop@smr.ru*. Области научных интересов: моделирование, разработка и исследование программных средств распределенной и параллельной обработки крупноформатных изображений; разработка алгоритмов и программного обеспечения систем технического зрения; разработка алгоритмов повышения качества цветных слабоконтрастных изображений.

Sergey Borisovich Popov, Candidate of Technical Sciense; Associate Professor of the Samara State Aerospace University; Senior researcher at the Image Processing Systems Institute of the RAS. E-mail: *spop@smr.ru*. His areas of research are parallel and distributed image processing, computer vision, color image processing.

Поступила в редакцию 9 июня 2010 г