Использование параллельных вычислений на графических процессорах AMD в задаче построения полей векторов перемещений

П. С. Любутин[1], С. В. Панин[1,2,*]

[1]Институт физики прочности и материаловедения СО РАН, Томск, Россия
[2]Национальный исследовательский Томский политехнический университет, Россия
[*]e-mail: svp@ispms.ru

Предложен алгоритм определения перемещений участков поверхности (оптического потока), основанный на их рекурсивном поиске, выполнение которого ориентировано на применение аппарата параллельных вычислений на графическом процессоре. Алгоритм параллельных вычислений реализован с использованием языка программирования OpenCL и ориентирован для исполнения на графических процессорах AMD Radeon. Результаты его тестирования показали, что применение графических процессоров позволяет кратно снизить время построения полей векторов перемещений при сохранении точности и помехоустойчивости расчета.

Ключевые слова: рекурсивный поиск перемещений, поле векторов перемещений, параллельный алгоритм, графический процессор.

Введение

Алгоритмы вычисления оптического потока (кажущегося поля движения) используются во многих научных и практических задачах, таких как оценка деформации материалов и элементов конструкций [1], анализ потоков жидкостей и газов [2], сжатие видеоданных [3], роботизированное управление автомобилями [4], детектирование движения в охранных системах и др. В задачах экспериментальной механики и технического зрения оптический поток определяется путем обработки серии изображений, отражающих перемещение объектов сцены относительно камеры либо, наоборот, наблюдателя относительно рабочей сцены.

Необходимость обработки видеоданных в режиме реального времени, а также постоянно возрастающее разрешение видеодатчиков требуют разработки быстродействующих алгоритмов и аппаратных средств для их выполнения. При этом снижение времени обработки данных может быть достигнуто за счет уменьшения вычислительной сложности алгоритмов либо применения принципиально новых аппаратных средств, в том числе многопроцессорных (многоядерных) вычислительных систем.

В настоящее время уровень развития вычислительных средств дает возможность использовать недорогие многоядерные аппаратные решения на основе графических процессоров, позволяющие задействовать аппарат параллельных вычислений и значительно снизить время обработки данных. Параллельные алгоритмы для графических процессоров применяются в различных задачах, связанных с оценкой перемещений.

Одной из них является слежение за объектами в режиме реального времени [5–7]. Алгоритмы вычисления поля движения (оптического потока) также активно реализуются с использованием параллельных вычислений [8–15]. В работе [8] авторами проведен обзор аппаратно-программных решений и вычислительных платформ для параллельных вычислений в задачах экспериментальной механики и оптических измерений величин перемещений.

В работе [9] описана реализация метода корреляции цифровых изображений с субпиксельной точностью расчета с применением программно-аппаратной архитектуры параллельных вычислений CUDA для исполнения на графических процессорах NVIDIA. В реализованном методе совместно используются обратный композиционный алгоритм Гаусса — Ньютона (IC-GN) для субпиксельной оценки перемещений и алгоритм кросс-корреляции, основанный на быстром преобразовании Фурье (БПФ) для оценки перемещений с дискретной (пиксельной) точностью.

Сравнение IC-GN алгоритма и прямого аддитивного алгоритма Ньютона — Рафсона проведено в [10]. Предложен параллельный алгоритм, основанный на определении положения "базовой" области изображения, относительно которой далее и определяются перемещения. Данные о ее деформации применяются далее для улучшения (повышения точности) начального приближения при определении перемещений соседних векторов, и каждый успешно рассчитанный вектор используется как новый, т. е. становится исходной информацией для проведения последующих вычислений.

Параллельная кратномасштабная схема определения поля перемещений предложена в [11]. Там же проведено ее сравнение с последовательной многомасштабной схемой. Отличительной особенностью схемы является то, что обработка изображений на каждом из масштабов производится параллельно в отличие от многомасштабного подхода. В рамках такой реализации на последнем этапе данные, полученные на каждом из предыдущих уровней, объединяются в один результирующий оптический поток.

В работе [12] предложен подход к параллелизации вычисления оптического потока видеофрагментов, снятых движущейся камерой в "неблагоприятных" условиях и естественном освещении. Съемка производилась с камеры, установленной на транспортное средство, передвигающееся по дорогам общего пользования. Для реализации выбран алгоритм Лукаса — Канаде. В [13] в качестве успешно реализованного подхода предлагается реализация параллельного алгоритма вычисления оптического потока, основанного на модифицированной версии нейронной сети типа самоорганизующейся карты для решения проблемы выбора размера апертуры. В [14] в качестве еще одного успешно решенного подхода представлена производительная реализация алгоритма для графического процессора GPU (Graphics Processing Unit) с целью вычисления оптического потока для последовательностей изображений, отражающих большие перемещения (Large Displacement Optical Flow — LDOF).

В работе [16] предложен алгоритм трехмерного рекурсивного поиска перемещений (3DRS), который позволил значительно снизить время обработки изображений. Алгоритм 3DRS используется для оценки полей векторов перемещений в системах обработки видеосигналов для применения в различных телевизионных устройствах. В работе [17] рассматривается задача реализации рекурсивного алгоритма для его выполнения на графическом процессоре с параллельной архитектурой. Алгоритм реализован с использованием программного интерфейса Direct3D и языка высокого уровня для программирования шейдеров HLSL [18, 19], а затем протестирован на видеокартах Nvidia GeForce 7800/8800.

В настоящей работе поставлена задача разработать параллельный алгоритм построения векторов перемещений на основе рекурсивного поиска (recursive search) с привлечением аппарата параллельных вычислений на графических процессорах и оценить достигаемое при этом снижение временных затрат.

1. Описание алгоритма

В предыдущей работе авторов [20] реализован быстродействующий алгоритм построения полей векторов перемещений, основанный на трехмерном рекурсивном поиске (3DRS). Алгоритм показал хорошую помехоустойчивость при малых вычислительных затратах. По этой причине в рамках разрабатываемого алгоритма авторы также поставили задачу задействовать 3DRS-подход. При этом в рамках "классического" исполнения 3DRS-алгоритма построение векторов производится по всему анализируемому изображению. Однако не всегда в задаче оценки деформации необходимо анализировать полноразмерное изображение, а можно задать размер и положение так называемой области интереса ROI (region of interest). По этой причине перед началом работы разрабатываемого алгоритма требуется задать размер, форму и положение области ROI. Кроме того, в приведенных в литературе данных авторы не нашли описания результатов реализации алгоритма 3DRS с применением аппарата параллельных вычислений с использованием графического процессора AMD Radeon и языка программирования OpenCL. Поэтому в рамках разрабатываемого алгоритма вычислений оптического потока было необходимо адаптировать "модифицированный" алгоритм 3-мерного рекурсивного поиска для его функционирования на графических процессорах AMD с использованием аппарата параллельных вычислений.

Алгоритм трехмерного рекурсивного поиска (3DRS) относится к методам блочной оценки перемещений, когда учитывается пространственная и временная связь векторов перемещений в пределах всего поля (оптического потока). На основе алгоритма трехмерного рекурсивного поиска, описанного в работе [16], реализован модифицированный алгоритм построения векторов перемещений, который также использует подход рекурсивного поиска перемещений (Displacements Recursive Search — DRS). Затем данный модифицированный алгоритм был переработан для проведения вычислений в параллельном виде (PDRS). Результатом работы алгоритма после обработки пары изображений является поле с регулярно расположенными векторами перемещений. Расстояние между центральными точками соседних блоков по вертикали и горизонтали задается такой величиной, как шаг *step* (рис. 1, *а*). Каждый вектор характеризует смещение участка изображения — блока — размером *bs* (рис. 1, *а*). Вычисление векторного поля производится построчно, блок за блоком, начиная с верхнего левого угла. Одна такая процедура называется проходом, а процесс построения поля — сканированием. При достижении последнего (правого нижнего) блока меняется направление сканирования, от правого нижнего блока к левому верхнему (и, соответственно, от нижней строки к верхней). Выполняется заданное количество проходов. При каждом последующем проходе векторы, построенные на предыдущих итерациях, могут быть скорректированы.

Алгоритм 3DRS основан на использовании векторов-кандидатов (рис. 1), которые задают предполагаемые направления перемещений. Для каждой площадки может быть использовано несколько таких векторов-кандидатов, включая пространственные и временные. В качестве пространственных кандидатов используются соседние векторы перемещений, которые были определены ранее. На рис. 1, *б* показаны четыре варианта

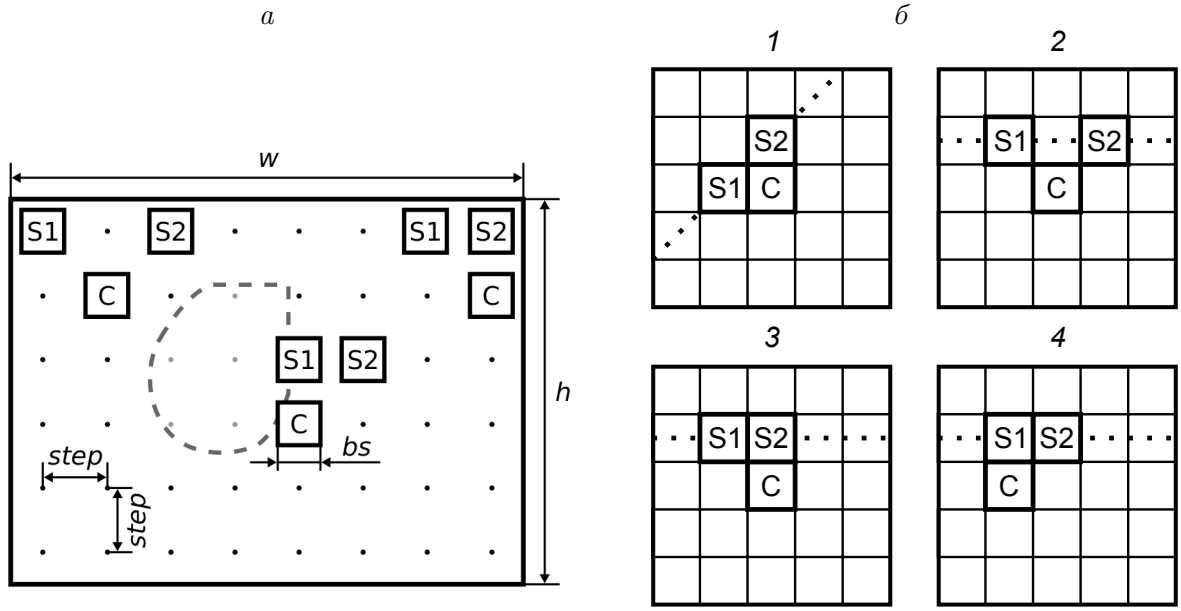

Рис. 1. Относительное расположение векторов-кандидатов: *а* — расположение с учетом границ векторного поля, *б* — варианты расположения блоков

относительного расположения векторов-кандидатов (обозначенных S1 и S2), где C — текущий блок, для которого оценивается смещение. В работе [17] описаны первые два из приведенных на рис. 1, *б* вариантов расчета по представленным схемам.

Одним из ограничений, возникающих при разработке параллельного алгоритма, является тот факт, что оценка перемещения каждого последующего блока зависит от результата определения предыдущих векторов-кандидатов. В связи с этим диагональное расположение векторов-кандидатов (вариант 1 на рис. 1, *б*) значительно усложняет алгоритм, в отличие от горизонтального расположения (вариант 2), поскольку количество блоков, лежащих на диагонали, изменяется. В то же время при расположении блоков по варианту 2 может быть реализована параллельная схема обработки данных, а именно поиск перемещений каждого блока в пределах одной строки векторного поля.

Помимо этого, в алгоритме 3DRS используются так называемые кандидаты обновления, которые генерируются из пространственных кандидатов S1 и S2 путем добавления случайных смещений. Генерация случайной последовательности неповторяющихся чисел в параллельном алгоритме сталкивается со сложностью практической реализации вследствие одновременного выполнения параллельных операций.

В реализованном авторами модифицированном алгоритме вместо случайных значений к каждому вектору-кандидату прибавляется смещение в диапазоне от -2 до 2 пикселов по вертикальному и горизонтальному направлениям. При отсутствии векторов-кандидатов (например, для первой строки) поиск смещения производится относительно перемещения самого блока. Для определения перемещения блока производится вычисление минимума меры подобия, в качестве которой используется сумма абсолютных разностей (Sum of Absolute Differences — SAD).

Решение задачи построения поля векторов перемещений на поверхности нагруженного материала допускает наличие разрывов, например концентраторов напряжений в виде отверстий или выточек либо конструктивных особенностей инженерных изделий. Исключение из обработки областей на изображении, соответствующих таким раз-

рывам, традиционно достигается путем предварительного задания области обработки на изображениях ROI. На рис. 1, *а* такая область вне изображения (снаружи) размером $w \times h$ ограничена его границами, а внутри (что, как правило, задается вручную путем наложения маски) — штриховой линией. Поскольку для блоков, находящихся у границы области обработки ROI, отсутствует правый либо левый вектор-кандидат, вариант 2 расположения кандидатов был дополнен вариантами 3 и 4. Наличие и относительное расположение векторов-кандидатов определяются для каждого блока до начала выполнения процедуры построения поля векторов перемещений. С этой целью формируются два массива, задающие относительное расположение векторов-кандидатов: один для прямого сканирования справа налево и сверху вниз, а второй для обратного направления (*Cand* на рис. 2).

Условные операции, предполагающие ветвление алгоритма, излишняя синхронизация между потоками в случае совместного использования доступа к памяти, использование векторных операций и небольшой объем локальной памяти вычислительных модулей графического процессора значительно ограничивают его возможности для выполнения параллельных вычислений. Кроме того, передача данных между потоками требует дополнительной синхронизации ядер графического процессора. С учетом особенностей предложенного DRS-алгоритма и вышеперечисленных ограничений, накла-

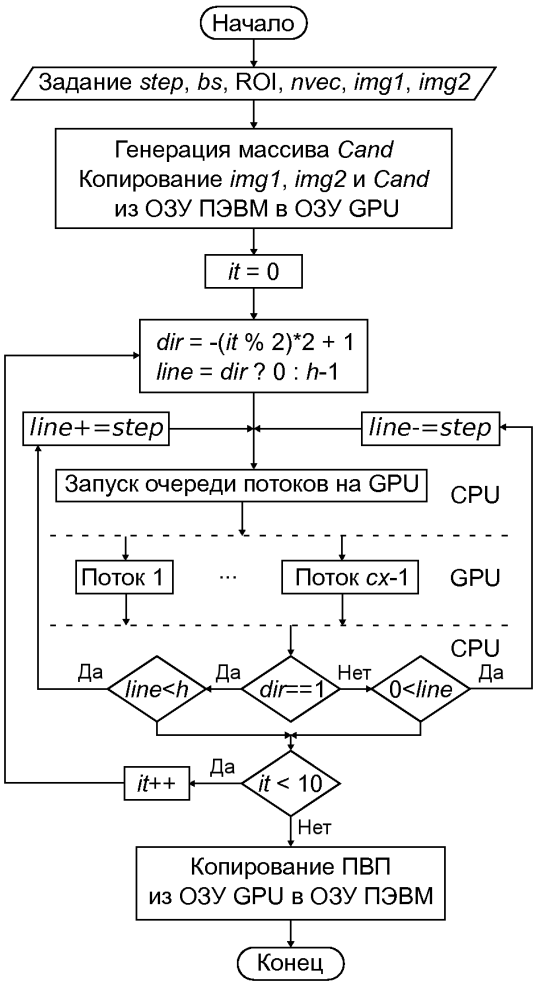

Рис. 2. Параллельный алгоритм определения полей векторов перемещений

дываемых аппаратом параллельных вычислений был разработан параллельный алгоритм построения полей векторов перемещений PDRS (рис. 2). Алгоритм реализован с использованием языка программирования OpenCL для выполнения на графических процессорах AMD Radeon.

Принцип организации вычислений на графическом процессоре может быть представлен следующим образом [21]. Прежде всего, проведение вычислений с применением программ, реализованных с использованием языка OpenCL как для центральных процессоров (CPU), так и для графических процессоров (GPU), принципиально отличается от таковых для программ, разрабатываемых на языках высокого уровня (таких как C, C++ и т. п.). OpenCL-приложение состоит из двух частей: ведущей программы и совокупности одного или нескольких кернелов (kernel) [21]. Кернел (фрагмент программы или ядро) представляет собой набор инструкций, написанных на языке OpenCL. При этом ведущая программа запускается на центральном процессоре. Кернелы запускаются на OpenCL-устройствах, например на графических процессорах. Ведущая программа вызывает команду, которая приводит в исполнение кернелы на OpenCL-устройстве. При запуске такая команда формирует пространство потоков, индексированных целыми числами. Копия кернела запускается для каждого потока, индексированного в таком пространстве. Каждая копия запущенного кернела называется рабочим элементом (work-item), который определяется координатами в этом индексированном пространстве. Эти координаты являются глобальным идентификатором для рабочего элемента. Затем команда, которая приводит в исполнение кернелы, создает набор рабочих элементов, каждый из которых использует одну и ту же последовательность инструкций.

Рабочие элементы объединены в рабочие группы (work-groups). Рабочие элементы имеют доступ к общей глобальной памяти, а также к локальной памяти. Каждой рабочей группе соответствует своя локальная память. Рабочим элементам, входящим в одну рабочую группу, доступна локальная память, которая недоступна элементам из другой рабочей группы. Глобальная память имеет объем порядка 1–4 ГБ. Локальная память ограничена размером 32 кБ на рабочую группу. Локальная память обладает значительно меньшим временем доступа, чем глобальная память, поэтому ее задействуют для хранения промежуточных результатов работы алгоритмов. Поскольку физически видеокарта представляет собой устройство, подключаемое к ПЭВМ по шине PCI Express, копирование данных из оперативной памяти ПЭВМ в память видеокарты занимает определенное время. Чтобы минимизировать потери времени, возникающие при выполнении операций копирования, данные копируются в память видеокарты один раз перед запуском кернелов (рис. 2). Память видеокарты (глобальная) есть не что иное, как глобальная память OpenCL-устройства, описанная выше. Выгрузка данных с видеокарты производится также один раз в конце работы программы. Входными данными для работы программы являются изображения (img1, img2) и параметры алгоритма, такие как шаг построения векторов (*step*), размер блока (*bs*), область обработки изображения (ROI), размерность векторных операций (*nvec*) и т. д. Выходными данными являются поле векторов перемещений и при необходимости величина минимальной меры SAD для каждого из блоков.

Предложенный алгоритм предполагает параллельное вычисление в пределах всей строки векторного поля. Каждая строка состоит из блоков, и для каждого из них определяется перемещение с учетом возможных 5×5 вариантов смещений (в диапазоне $-2...2$) для каждого вектора-кандидата. При этом максимальное количество векторов-кандидатов равно 2. С учетом этого размер рабочей группы (work-group) определялся

как $5 \times 5 \times 2 = 50$, т. е. размер локальных и глобального массивов рабочих элементов для параллельной обработки localSize и globalSize определяли следующим образом:

$$globalSize[0] = (2d+1)2cx; \quad globalSize[1] = 2d+1;$$
$$localSize[0] = (2d+1)2cx; \quad localSize[1] = 2d+1,$$

где d — максимальное смещение и cx — размер строки векторного поля (количество блоков). Взаимодействие между ведущим устройством (ПЭВМ) и OpenCL-устройством выполняется с помощью очереди команд, отправляемых от ПЭВМ. Очереди команд делятся на очереди кернелов, очереди работы с памятью и очереди синхронизации. За один запуск очереди кернелов, определяемых через localSize и globalSize в функции OpenCL clEnqueueNDRangeKernel, строится одна строка поля векторов перемещений. Операция вычисления меры подобия SAD выполняется одним рабочим элементом. Кроме непосредственно вычисления меры SAD кернел выполняет поиск минимального значения SAD из 50 элементов массива для соответствующих смещений. При этом процедура поиска минимума SAD выполняется только рабочим элементом рабочей группы с локальными индексами $[0, 0]$.

Видеокарты семейства AMD Radeon поддерживают различные операции над векторными данными. Использование векторных операций позволяет значительно уменьшить время обработки изображений. Наиболее ресурсоемкая задача в реализованном алгоритме построения полей векторов перемещений — это вычисление меры подобия SAD, для расчета которой были задействованы векторные операции. Для этого предложено привести указатели на области памяти с изображениями, имеющими скалярный тип, к указателям векторного типа. При этом изображения имели полутоновой формат представления (grayscale). Использование векторного представления int2 и int4 накладывает следующее ограничение: позиция, с которой производится чтение данных векторного типа, должна быть выровнена кратно размеру вектора (2 или 4). Было решено на каждое изображение выделить блок памяти, равный размеру изображения, кратный размеру вектора. Далее память последовательно (друг за другом) заполняется копиями изображений по размеру вектора, при этом каждая копия смещена по оси x на величину от нуля до размера вектора, уменьшенного на единицу. Таким образом, были реализованы три кернела, использующие для вычисления меры SAD-операции над типами данных int, int2 и int4. Требование к дополнительному расходу памяти является недостатком данного решения. Несмотря на это, как показало тестирование, время обработки изображений было существенно снижено.

2. Тестирование

В работе применяли рабочую станцию, оснащенную процессором Intel Core i7-3770K и графической картой AMD Radeon HD 7970, а также ПЭВМ, оснащенную графической картой Radeon HD 5570 и процессором Intel Core i5-3470 (для целей сравнения). Программная реализация алгоритмов параллельных вычислений осуществлялась с использованием открытого языка программирования OpenCL.

Тестирование алгоритмов проводили на цифровых оптических изображениях с разрешением 3456×5184. Использовали изображения поверхности образцов углерод-углеродного композиционного материала с концентратором напряжений в виде центрального отверстия, полученных в ходе испытаний на статическое растяжение. Для регист-

рации изображений использовали цифровую зеркальную фотокамеру Canon 550D, оснащенную телеобъективом.

Размер блока изображения при определении перемещений задавался равным 64×64 пиксела, шаг между векторами 48 пикселов. В алгоритме использовалось 10 итераций сканирования. Для построения полей векторов перемещений на центральных процессорах Intel Core i5 и i7 алгоритм реализован на языке программирования C++, было использовано одно ядро процессора. Результаты тестирования показали, что применение графических процессоров позволяет значительно снизить время вычисления полей векторов перемещений (рис. 3, *а*). Снижение времени при заданных параметрах расчета и для данных изображений составило 27 раз. При увеличении плотности векторного поля в 3 раза (шаг между векторами равен 16 пикселам) и соответственно количества векторов в строках временные затраты при использовании параллельного алгоритма были снижены в 63 раза по сравнению с чисто программной реализацией алгоритма

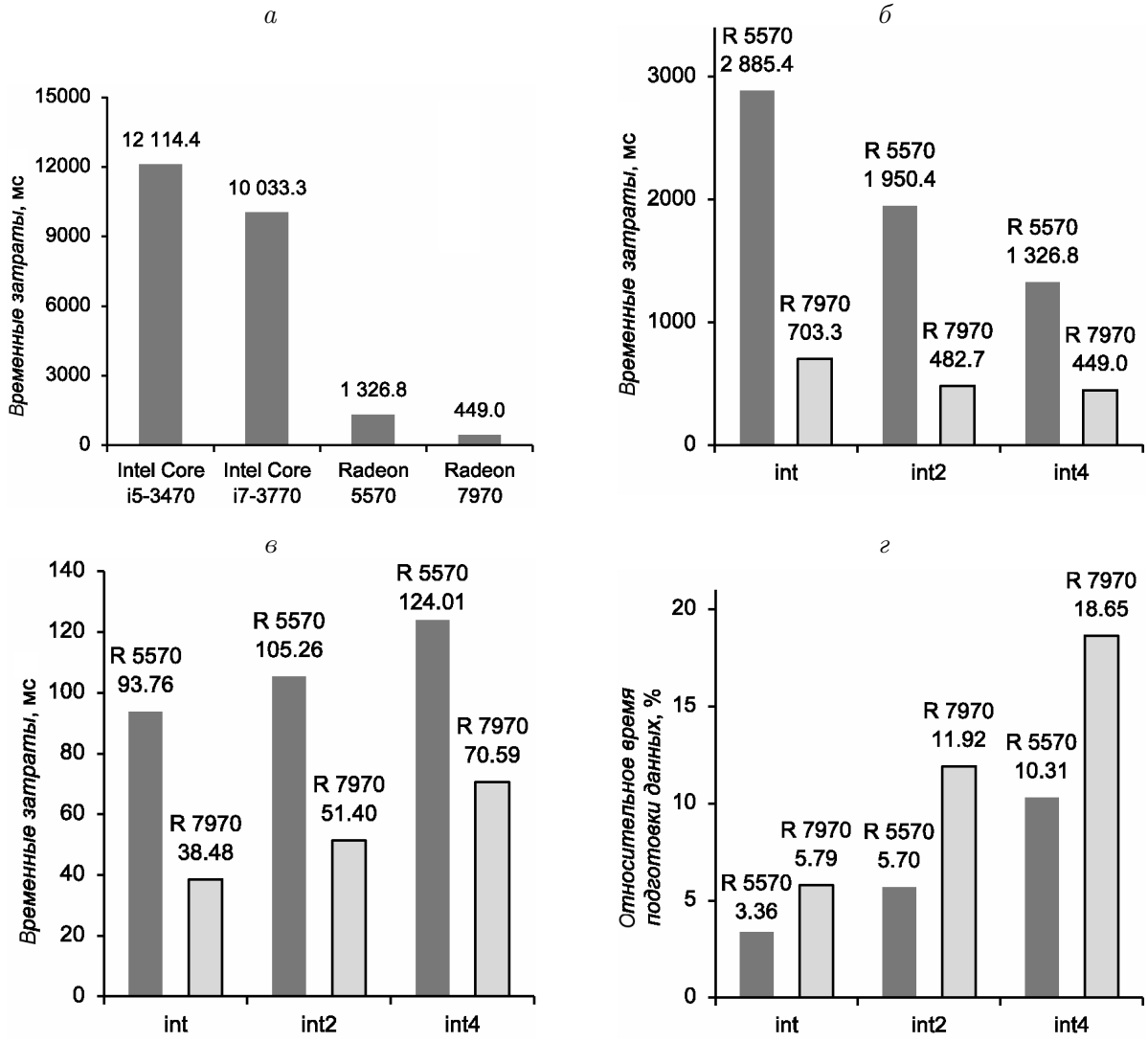

Рис. 3. Временные затраты различных операций при построении полей векторов перемещений: время построения полного векторного поля (*а*); то же с применением различных векторных операций (*б*); время, затрачиваемое на подготовку данных (*в*); отношение временных затрат на подготовку данных к времени их обработки (*г*)

DRS. Это связано с тем, что программа задействует большее количество ядер видеокарты при параллельном расчете перемещений в строке.

Применение векторных операций позволило снизить временные затраты, причем это особенно актуально для видеокарты предыдущего поколения Radeon 5570 (рис. 3, *б*). Чем больше размер типа векторных данных, тем эффективнее используется графический процессор. Такое их поведение связано с архитектурными особенностями. Поскольку для задействования векторных операций при использовании разработанного алгоритма необходимо выделять больший объем памяти и копировать больше данных, было крайне важно оценить время, затрачиваемое на операции копирования, предшествующие построению поля векторов перемещений. На рис. 3, *в* видно, что временные затраты на подготовку (копирование) данных растут с увеличением размеров типов векторных данных. Однако вклад операций по выделению памяти и копированию данных не приводит к значительному увеличению временных затрат. Также была проведена оценка отношения временных затрат на подготовку данных ко времени их обработки при различных размерах типов векторных данных (рис. 3, *г*). Из диаграммы видно, что относительное время подготовки данных растет при увеличении размеров типов векторных данных. Для графического процессора Radeon 7970 отношение времени, затраченного на подготовку данных, ко времени обработки данных больше, чем для Radeon 5570. При использовании типа int4 такое отношение достигает почти 20 %, что составляет значительную часть времени работы алгоритма.

Выводы

На основе метода трехмерного рекурсивного поиска (3DRS) реализован и протестирован модифицированный алгоритм определения перемещений участков изображений — DRS. Для его выполнения на графических процессорах AMD Radeon разработан параллельный алгоритм построения полей векторов перемещений PDRS, который реализован с использованием языка программирования OpenCL. При тестировании указанных алгоритмов получены следующие результаты.

1. Применение графических процессоров позволяет значительно снизить время построения полей векторов перемещений. В частности, при заданных параметрах расчета при обработке изображений разрешением 3456×5184 снижение времени составило 27 раз.

2. При увеличении плотности векторного поля в 3 раза (при размере шага между векторами 16 пикселов) и соответствующем количестве векторов в строках временные затраты при использовании параллельного алгоритма PDRS были снижены в 63 раза по сравнению с алгоритмом DRS, реализованным на языке высокого уровня C++ для центрального процессора. Достигнутый результат обусловлен тем, что при увеличении размера строки векторного поля программа задействует большее количество ядер видеокарты при параллельном способе построения векторов в пределах этой строки.

3. Для вычисления меры подобия SAD применены векторные операции, что позволило снизить временные затраты в 1.5–2 раза в случае применения видеокарт AMD Radeon 5570 и 7970.

4. Использование векторных операций на видеокарте предыдущего поколения Radeon 5570 обусловило больший эффект снижения вычислительных затрат по сравнению с более современной Radeon 7970 (при одной и той же реализации алгоритма PDRS), что связано с отличиями архитектуры двух графических процессоров.

Благодарности. Работа выполнена в рамках Программы фундаментальных научных исследований государственных академий наук на 2013–2020 гг.

Список литературы / References

[1] **Schreier, H., Orteu, J.-J., Sutton, M.A.** Image correlation for shape, motion and deformation measurements: Basic concepts, theory and applications. Springer, 2009. 321 p.

[2] **Raffel, M., Willert, C.E., Wereley, S.T., Kompenhans J.** Particle image velocimetry. A practical guide. 2nd ed. Berlin; Heidelberg: Springer, 2007. 448 p.

[3] **Беляев Е.А., Тюрликов А.М.** Алгоритмы оценки движения в задачах сжатия видеоинформации на низких битовых скоростях // Компьютерная оптика. 2008. Т. 32, № 4. С. 403–412.
Belyaev, E.A., Tyurlikov, A.M. Motion estimation algorithms for low bit-rate video Compression // Computer Optics. 2008. Vol. 32, No. 4. P. 403–412. (In Russ.)

[4] **Giachetti, A., Campani, M., Torre, V.** The use of optical flow for road navigation // IEEE Transactions on Robotics and Automation. 1998. Vol. 14, No. 1. P. 34–48.

[5] **Doyle, D.D., Jennings, A.L., Black, J.T.** Optical flow background estimation for real-time pan/tilt camera object tracking // Measurement. 2014. Vol. 48. P. 195–207.

[6] **Garrigues, M., Manzanera, A.** Real time semi-dense point tracking // Image Analysis and Recognition: 9th Intern. Conf., ICIAR 2012. Aveiro, Portugal, June 25–27, 2012. Proceedings. Pt I. Berlin; Heidelberg: Springer, 2012. P. 245–252.

[7] **Mahmoudi, S.A., Kierzynka, M., Manneback, P., Kurowski, K.** Real-time motion tracking using optical flow on multiple GPUs // Bulletin of the Polish Academy of Sciences: Technical Sciences. 2014. Vol. 62, No. 1. P. 139–150.

[8] **Gao, W., Kemao, Q.** Parallel computing in experimental mechanics and optical measurement: A review // Optics and Lasers in Engineering. 2011. Vol. 50, No. 4. P. 608–617.

[9] **Zhang, L., Wang, T., Jiang, Z., Kemao, Q., Liu, Y., Liu, Z., Tang, L., Dong, S.** High accuracy digital image correlation powered by GPU-based parallel computing // Optics and Lasers in Engineering. 2015. Vol. 69. P. 7–12.

[10] **Shao, X., Dai, X., He, X.** Noise robustness and parallel computation of the inverse compositional Gauss–Newton algorithm in digital image correlation // Optics and Lasers in Engineering. 2015. Vol. 71. P. 9–19.

[11] **Barranco, F., Díaz, J., Pino, B., Ros, E.** A multi-resolution approach for massively-parallel hardware-friendly optical flow estimation // J. of Visual Communication and Image Representation. 2012. Vol. 23, No. 8. P. 1272–1283.

[12] **Garcia-Dopico, A., Pedraza, J.L., Nieto, M., Pérez, A., Rodríguez, S., Navas, J.** Parallelization of the optical flow computation in sequences from moving cameras // Eurasip J. on Image and Video Processing. 2014. DOI:10.1186/1687-5281-2014-18. Available at: http://jivp.eurasipjournals.springeropen.com/articles/10.1186/1687-5281-2014-18

[13] **Shiralkar, M.P., Schalkoff, R.J.** A self-organization based optical flow estimator with GPU implementation // Machine Vision and Applications. 2012. Vol. 23, No. 6. P. 1229–1242.

[14] **Sundaram, N., Brox, T., Keutzer, K.** Dense point trajectories by GPU-accelerated large displacement optical flow // Computer Vision — ECCV 2010. Proc. 11th European Conf. on Computer Vision. Pt 1. Heraklion, Crete, Greece, September 5–11, 2010. Lecture Notes in Computer Science. 2010. Vol. 6311. P. 438–451.

[15] **Plyer, A., Le Besnerais, G., Champagnat, F.** Massively parallel Lucas Kanade optical flow for real-time video processing applications // J. of Real-Time Image Proc. 2016. Vol. 11, No. 4. P. 713–730.

[16] **De Haan, G., Biezen, P.W.A.C., Huijgen, H., Ojo, O.A.** True-motion estimation with 3-D recursive search block matching // IEEE Transactions on Circuits and Systems for Video Technology. 1993. Vol. 3, No. 5. P. 368–379.

[17] **Zhao, M., van der Heijden, H.** 3D recursive search block matching on graphics processing unit // Consumer Electronics, 2008. ICCE 2008. Digest of Technical Papers. Intern. Conf. 9–13 Jan. 2008. DOI:10.1109/ICCE.2008.4587939. Available at: http://ieeexplore.ieee.org/document/4587939/?arnumber=4587939

[18] **Gray, K.** Microsoft DirectX 9 programmable graphics pipeline. Microsoft Press, 2003. 496 p.

[19] **St-Laurent, S., Engel, W.** The complete effect and HLSL guide. London: Paradoxal Press, 2005. 324 p.

[20] **Панин С.В., Титков В.В., Любутин П.С.** Снижение вычислительных затрат с применением алгоритма трехмерного рекурсивного поиска при построении векторов перемещений в оптическом методе оценки деформации // Вычисл. технологии. 2013. Т. 18, № 5. С. 91–101.
Panin, S.V., Titkov, V.V., Lyubutin, P.S. Computationally effective three dimensional recursive algorithm for calculation of the translation vectors in the optical method for assessment of deformations // Comput. Technologies. 2013. Vol. 18, No. 5. P. 91–101. (In Russ.)

[21] **Munshi, A., Gaster, B., Mattson, T., Fung, J., Ginsburg, D.** OpenCL programming guide. Addison-Wesley Professional, 2011. 648 p.

Поступила в редакцию 27 марта 2015 г.,
с доработки — 9 сентября 2016 г.

The use of parallel computing with the AMD graphics processors for the construction of displacement vector fields

Lyubutin, Pavel S.[1], Panin, Sergey V.[1,2,*]

[1]Institute of Strength Physics and Materials Science, SB RAS, Tomsk, 634055, Russia
[2]National Research Tomsk Polytechnic University, Tomsk, 634050, Russia

*Corresponding author: Panin, Sergey V., e-mail: svp@ispms.ru

The aim of the study is to develop a parallel algorithm for constructing displacement vector fields based on recursive search approach with the use of parallel computing at GPU and estimation of time costs. A modified algorithm for the displacement estimation DRS (direct recursive search) was implemented beingand based on the approach to three-dimensional recursive search (3DRS). By considering the DRS features and limitations for the hardware parallel computing, the parallel algorithm PDRS for constructing displacement vector fields has been developed. The latter employs recursive search, which was implemented with the use of the OpenCL programming language to run onto GPUs AMD Radeon.

Test results showedhave shown that the use of graphics processors can significantly reduce the time of construction of displacement vector fields. Processing time at preset

parameters and images with a resolution of 3456×5184 was decreasing by ~ 27 times. When the density of the vector field was increased by three times (spacing between vectors was equal to 16 pixels) as well as the number of vectors in the lines the time required for implementing the parallel algorithm PDRS was 63 times less as compared to the DRS algorithm, realized withwhich was implemented using the high level language C++ for the CPU operation by CPU. This is related to the fact that with increasing the size of the vectors in the line the program uses a larger number of cores in parallel graphics method for constructing vectors in a row as the size of the vectors in the line increases.

The vector operations were applied to compute the similarity measure SAD. Their employment incorporation allows reducing the computation time by 1.5–2 times at use offor AMD Radeon 5570 and 7970 graphical cards. In contrast with, the current implementation being realized with, which uses the PDRS algorithm that employs vector operations on the previous generation graphics card Radeon 5570 a previous generation one shows a has shown greater effect of reducing the computation time. The reason is associated with the differences between the architecture of those two video processors.

Keywords: recursive displacements search, displacement vector field, parallel algorithm, graphics processing unit.

Acknowledgements. The research was performed within the Programs for Fundamental Studies of fundamental studies of the State Academies of Sciences (2013–2020).

Received 27 March 2015
Received in revised form 9 September 2016