Массивно-многопоточная реализация двумерных БИХ фильтров

А.В. Никоноров[1], В.А. Фурсов[1], П.Ю. Якимов[2]

Институ систем обработки изображений РАН[1], Самарский Государственный Аэрокосмический Университет имени академика С.П. Королева[2]

В работе рассматривается информационная технология реализации двумерного фильтра с бесконечной импульсной характеристикой (БИХ-фильтра). Для обеспечения физической реализуемости БИХ-фильтр строится в виде параллельного соединения 4-х фильтров одного квадранта. Параметры каждого фильтра одного квадранта определяются путем идентификации по тестовым фрагментам, формируемым из исходного искаженного изображения. После определения параметров фильтра осуществляется анализ его устойчивости. Для обработки крупноформатного изображения с использованием построенных указанным способом четырех фильтров одного квадранта предлагается схема эффективной массивно-многопоточной реализации БИХ фильтра в системах на основе графических процессоров (GPU).

1. Введение

В работе [1] рассматривалась информационная технология определения характеристик и обработки изображений с использованием двумерных фильтров с бесконечной импульсной характеристикой (БИХ-фильтров). В частности, для решения задачи идентификации параметров фильтра в указанной работе предложено использовать малые тестовые фрагменты, которые формируются из искаженного изображения с использованием априорной информации о геометрической форме регистрируемых объектов. При этом двумерный БИХ-фильтр строится в виде параллельного соединения физически реализуемых фильтров с опорной областью в виде квадранта, обеспечивающих компенсацию сильных искажений с использованием опорной области небольших размеров.

Вопросы повышения производительности обработки одномерными БИХ фильтрами за счет использования параллельных вычислительных ресурсов достаточно хорошо исследованы. В работе [2] рассматривалась реализация одномерных БИХ фильтров в многопоточных вычислительных средах. В работе [3] приведены результаты GPU-реализации БИХ фильтра. В ряде случаев возможна декомпозиция обработки изображений, в результате которой задачи сводится к последовательному применению одномерных фильтров [4]. Однако, в общем случае, такую декомпозицию провести нельзя.

В настоящей работе рассматривается достаточно универсальная массивно-многопоточная реализация двумерных БИХ фильтров, с использованием графических процессоров и технологии CUDA, обеспечивающая значительное увеличение производительности по сравнению не только с CPU-процессорами, но и с обычными CUDA-процедурами, использующими неоптимизированные коды.

2. Технология формирования и идентификации двумерного БИХ-фильтра

В общем случае выражение, определяющее способ вычисления выходного отсчета $y(n_1, n_2)$ двумерного БИХ-фильтра, имеет вид [5]:

$$y(n_1,n_2)=\sum_{r_1}\sum_{r_2} a(n_1-r_1,n_2-r_2)x(r_1,r_2)-\sum_{\substack{k_1\\(k_1k_2\neq\\ n_1n_2)}}\sum_{k_2} b(n_1-k_1,n_2-k_2)y(k_1,k_2). \qquad (1)$$

Нетрудно заметить, что в данном случае требование рекурсивной вычислимости не выполняется (вычисляемый выходной отсчет на рисунке 1, а обозначен кружком в центре опорной области). Для физической реализуемости двумерного БИХ-фильтра он представляется в виде параллельного соединения БИХ-фильтров одного квадранта [6] (рисунок 1, б-д).

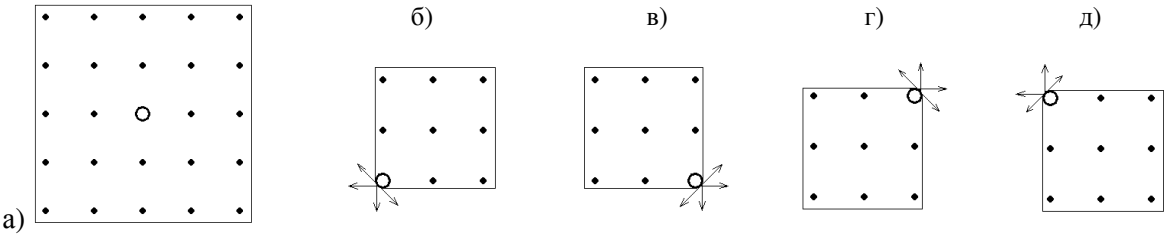


Рис. 1. Исходная маска 5×5 – а) и соответствующие ей маски и допустимые направления рекурсии для: 1-го квадранта – б); 2-го квадранта – в); 3-го квадранта – г); 4-го квадранта – д).

Для заданной опорной маски в виде квадранта и некоторой пары тестовых фрагментов, один из которых сформирован путем компьютерного ретуширования, записывается систему линейных уравнений [1]:

$$\mathbf{Y} = \mathbf{S}\varphi + \mathbf{o}, \qquad (2)$$

где, в соответствии с (1)

$$\mathbf{Y} = \begin{bmatrix} y_1(n_1,n_2) \\ y_2(n_1,n_2) \\ \vdots \\ y_N(n_1,n_2) \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} x_1(r_1,r_2) & \cdots & y_1(k_1,k_2) & \cdots \\ x_2(r_1,r_2) & \cdots & y_2(k_1,k_2) & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ x_N(r_1,r_2) & \cdots & y_N(k_1,k_2) & \cdots \end{bmatrix}, \mathbf{o} = \begin{bmatrix} \xi_1(n_1,n_2) \\ \xi_2(n_1,n_2) \\ \vdots \\ \xi_N(n_1,n_2) \end{bmatrix},$$

N – число отсчетов на тестовых фрагментах, по которым сформирована система (2), $\varphi = \left[a(n_1 - r_1, n_2 - r_2), \ \cdots \ b(n_1 - k_1, n_2 - k_2), \ \cdots \right]^T$ - искомый вектор параметров фильтра.

Для обеспечения возможности идентификации фильтра высокого порядка в работе [1] матрицу **S** в (5) предложено составлять из блочных матриц, каждая из которых формируется по простым фрагментам, на каждом из которых функция яркости изменяется в одном направлении, но эти направления для разных фрагментов различны. Примеры таких фрагментов приведены в работе [1].

Для вычисления параметров фильтра по данным системы (2) может использоваться простейшая оценка метода наименьших квадратов (МНК):

$$\hat{\varphi} = \left[\mathbf{S}^T\mathbf{S}\right]^{-1}\mathbf{S}^T\mathbf{Y}, \qquad (3)$$

либо более устойчивая к грубым ошибкам типа сбоев оценка метода наименьших модулей (МНМ-оценка).

3. Анализ устойчивости фильтра

Анализ устойчивости фильтра необходим вследствие того, что получающиеся в результате идентификации фильтры одного квадранта могут оказаться неустойчивыми, что может привести к искажениям обработанного изображения [6]. Поэтому в качестве промежуточного этапа технологии предлагается осуществлять анализ устойчивости полученного фильтра.

Передаточная функция БИХ-фильтра, который мы идентифицируем, выглядит следующим образом:

$$H_z(z_1, z_2) = \frac{\sum_{r_1=0}^{N_1}\sum_{r_2=0}^{N_2} a(r_1,r_2)\, z_1^{-r_1} z_2^{-r_2}}{\sum_{k_1=0}^{N_1}\sum_{k_2=0}^{N_2} b(k_1,k_2)\, z_1^{-k_1} z_2^{-k_2}}, \qquad (4)$$

где N_1, N_2 – это размеры входной маски (будем считать, что выходная маска имеет такие же размеры, что и входная).

Нам необходимо исследовать поведение функции, находящейся в знаменателе. Для удобства обозначим:

$$B(z_1, z_2) = \sum_{k_1=0}^{N_1} \sum_{k_2=0}^{N_2} b(k_1, k_2) z_1^{-k_1} z_2^{-k_2} .$$

В качестве основы для проведения анализа устойчивости целесообразно опираться на теорему Шэнкса [7]. В соответствии с этой теоремой для каждой точки $b = |z_2|$ единичной окружности $|z_2| = 1$ на комплексной плоскости z_2 для всех $|z_1| \geq 1$ должно выполняться условие:

$$B(z_1, b) \neq 0,$$

а для каждой точки $a = |z_1|$ единичной окружности $|z_1| = 1$ на комплексной плоскости z_1 для всех $|z_2| \geq 1$ должно выполняться условие:

$$B(a, z_2) \neq 0.$$

Для проверки выполнения первого из указанных условий строится годограф корней характеристического уравнения

$$B(z_1, b) = 0$$

на плоскости z_1 при условии, что параметр $b = |z_2|$ «пробегает» все точки на единичной окружности в плоскости z_2. Если этот годограф пересекает единичную окружность на плоскости z_1, это свидетельствует о неустойчивости фильтра. Устойчиваость возможна только в случае, если пересечения годографов с единичной окружностью отсутствуют, и все они оказываются внутри круга единичного радиуса плоскости z_1.

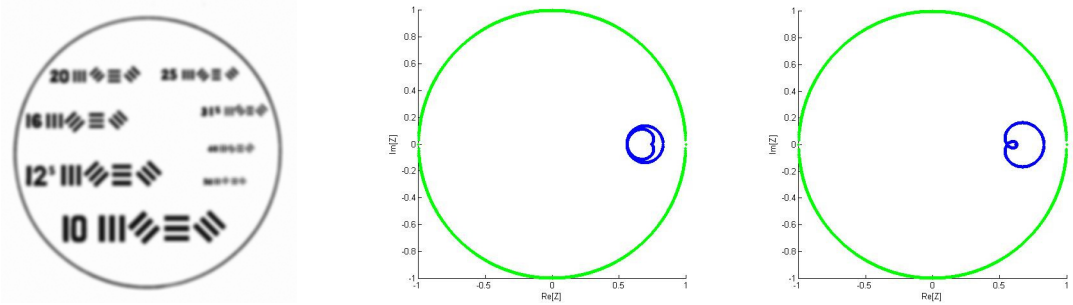
Аналогично годограф корней характеристического уравнения

$$B(a, z_2) = 0$$

на плоскости z_2 образованный при условии, что параметр $a = |z_1|$ «пробегает» все точки на единичной окружности в плоскости z_1, должен находиться внутри круга единичного радиуса на плоскости z_2.

При проведении анализа устойчивости следует иметь в виду, что числа *a* и *b* в общем случае являются комплексными, поэтому при вычислении корней следует приравнивать нулю как действительную, так и мнимую часть полинома в левой части характеристического уравнения.

На рисунках 2,б и 2,в показано поведение годографа устойчивого фильтра первого квадранта, идентифицированного с использованием миры, показанной на рисунке 2,а.



а) б) в)

Рис. 2. а) Мира, использованная для идентификации параметров БИХ-фильтра
б) Годограф корней характеристического уравнения $B(z_1, b) = 0$
в) Годограф корней характеристического уравнения $B(a, z_2) = 0$

Из рисунка 2 можно заключить, что исследуемый БИХ-фильтр, полученный идентификацией из тестовых фрагментов, является устойчивым. На рисунке хорошо видно, как годографы корней уравнения $B(z_1, z_2) = 0$ хорошо вписываются в единичную окружность.

4. Массивно-многопоточная реализация БИХ-фильтра в GPU системе

Поквадрантная реализация БИХ фильтра [1] позволяет естественным образом провести распределение вычислений на четыре потока. В настоящее работе предлагается массивно-многопоточных алгоритм реализации БИХ фильтра. Такая реализация в первую очередь ориентирована на использование в GPGPU системах, однако также может быть использована в современных многоядерных CPU системах.

В работе [9] была предложена процедура эффективной реализации рекуррентной обработки изображения локальным окном. С небольшими изменениями эта процедура может быть использована для решения задачи двумерной КИХ фильтрации.

В настоящей работе предлагается эффективная процедура реализации квадрантного БИХ фильтра. БИХ фильтр можно представить в виде последовательного соединения двух фильтров-компонент. Первый КИХ компонент зависит только от входного сигнала и соответствует числителю формулы (4). Второй, зависит только от выхода и соответствует знаменателю в формуле (4). Будем называть его рекурсивным компонентом фильтра.

Рассмотрим реализацию рекурсивного компонента. Будем рассматривать один квадрант, (x>0, y>0), для остальных трех квадрантов процедура выполняется аналогично. Передаточная функция для такого компонента имеет вид:

$$\frac{1}{\sum_{i=0,j=0}^{m-1} b_{i,j} z_1^{-i} z_1^{-j}}, \quad b_{0,0} = 1 \cdot$$

Схема вычислений описывается далее в терминах потоков (thread), блоков (block) и сетки (grid). Эти термины стандартны для архитектур CUDA и OpenCL [8, 10]. Предлагаемая схема содержит два уровня параллелизма – на уровне блоков и на уровне потоков.

Каждый блок выполняет расчет для одного столбца данных. Для единообразия, полагаем, что номер блока равен номеру столбца данных, т.е. блок с индексом *g* должен *N* раз выполнить следующее суммирование:

$$y_{k,g} = \sum_{i=0,j=0}^{m-1} b_{i,j} y_{k-i,g-j}, \quad k = 0,1,...N. \tag{5}$$

Так как вычисления выполняются рекурсивно, блоку для вычисления значения в некоторой точке (k,t) необходимо чтобы блоки с номерами меньше j выполнили вычисления для не менее чем i строк. Таким образом, если все блоки начинают вычисления одновременно, блок с индексом *b* должен дождаться завершения вычислений предыдущими блоками.

Потоки каждого блока выполняют суммирование (5) параллельно, по схеме каскадного суммирования или редукции[8, 10]. В таком суммировании должно быть задействовано $2^{2]\log_2 m[}$ потоков и тогда количество операций выполняемых потоками параллельно составит

$$O_{cs} = 2]\log_2 m[, \tag{6}$$

где под]m[понимается наименьшее целое, большее m.

Тогда количество вычислений для каждого блока составит:

$$O_{bs} = 2N]\log_2 m[. \tag{7}$$

Так как количество (6) для всех блоков одинаковое и блок с индексом *g* должен ожидать завершения вычисления для (*g*-1) блоков, то время ожидания для блока составит примерно

$$O_{bw} = 2(g-1)]\log_2 m[. \tag{8}$$

Если предположить, что N блоков начинают вычисления одновременно, то для последнего блока, с учетом (7) и (8), количество операций составит

$$O = (4N - 2)]\log_2 m[\,. \qquad (9)$$

Количество операций, требуемое для последовательной реализации алгоритма, составляет $O_S = N^2 m^2$. Таким образом, теоретическая оценка ускорения параллельной реализации позволяет предположить значительное ускорение. Однако модель многопоточности современных GPU накладывает некоторые дополнительные условия на реализацию предлагаемого алгоритма.

В общем случае, если количество блоков равно G, то алгоритм выполняется полосами]N/G[раз, а суммарная сложность должна составить:

$$O =]N / G[(4N - 2)]\log_2 m[\,.$$

Зависимость (7) имеет квадратичный характер по N, что должно сказываться на производительности при G << N.

Возможность увеличения количества блоков G зависит от возможностей GPU. В настоящей работе проведено исследование производительности при увеличении G для видеоадаптера GF 9600. Время обработки для одной полосы и для всего изображения для N = 512, m = 5 и числе потоков 16 для различных значений G приведено на рисунке 3.

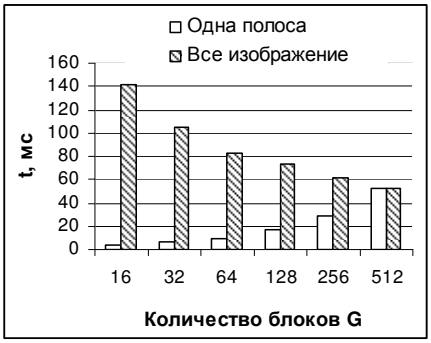


Рис. 3. Зависимость времени обработки от количества блоков.

Увеличение количества блоков приводит к увеличению накладных расходов планировщика потоков. Однако планировщик GPU исключает тупиковые ситуации, и увеличение числа блоков позволяет увеличить производительность.

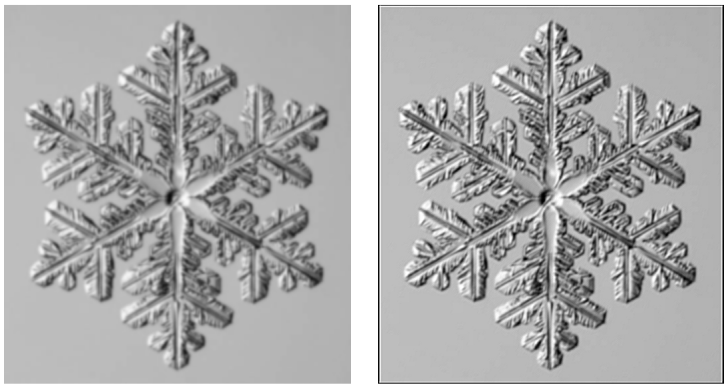
5. Примеры обработки

Экспериментальные исследования качества предложенной технологии проводились в следующей последовательности.

- Автоматизированный поиск тестовых фрагментов.
- Ретушь найденных тестовых фрагментов.
- Идентификация параметров БИХ-фильтров по тестовым фрагментам.
- Проверка на устойчивость.
- Обработка изображений с помощью полученных квадрантных БИХ-фильтров.

В качестве тестового изображения использованы снимки кристаллов льда. В результате все идентифицированные фильтры получились устойчивыми, что позволило получить значительное улучшение обработанных изображений.

На рисунке 4 приведен пример обработки изображений с использованием описанной технологии. а) –«размытое» изображение, б) – обработанное.



а) размытое изображение б) обработанное изображение

Рис. 4. Результат обработки изображения кристалла льда.

6. Заключение

Рассмотренная технология обработки изображений с использованием БИХ-фильтров является существеным развитием технологии описанной в работе [1]. Включение в качестве одного из этапов технологии проверки устойчивости полученного в результате идентификации БИХ-фильтра обеспечивает повышение качества и существенное повышение надежности обработки крупноформатных изображений.

Наиболее важным результатом работы является рекуррентная реализация БИХ-фильтра на GPU-процессорах. В данном случае путем оргинальной организации процедур обработки удалось преодолеть традиционно существовавшее мнение, что эффективная реализация рекурсивных процедур на GPU-процессорах невозможна. Также интересным для дальнейших исследований является анализ эффективности параллельной реализации БИХ-фильтров в RDMA системах.

Литература

1. Милюткин М.Г., Никоноров А.В., Фурсов В.А., Параллельная реализация двумерных БИХ-фильтров в распределенной системе обработки изображений, Труды международной конференции ПаВТ 2010, 2010 г., 268-275.
2. W. Sung and S. K. Mitra, Effiicient Multi-Proccessor Implementtation of Recursive Digital Filters, ICASSP, 1986.
3. Mccool M.D., Signal Processing and General-Purpose Computing and GPUs, Signal Processing Magazine, IEEE, 2008, pp. 109-114.
4. Мурызин С.А., Сергеев В.В., Фролова Л.Г. Исследование эффективности двумерных параллельно-рекурсивных КИХ-фильтров // Компьютерная оптика. - М.: МЦНТИ, 1992. - Вып.12. - С.65-71.
5. Методы компьютерной обработки изображений / Под ред. Сойфера В.А., Москва, Физматлит, 2001.
6. Зимин Д.И., Фурсов В.А. Построение устойчивых алгоритмов обработки изображений путем аппроксимации фильтров с бесконечной импульсной характеристикой // сб. Компьютерная оптика, № 28, 2005, с. 124-127.
7. D. E. Dudgeon, R. M. Mersereau, Multidimensional digital signal processing, Prentice-Hall, Inc., Englewood Cliffs, 1984.
8. А. В. Боресков, А. А. Харламов, Основы работы с технологией CUDA, ДМК Пресс, 2010, 230 с.
9. S. Bibikov, V. Fursov, A. Nikonorov, P. Yakimov, Memory Optimization for Recurrent CUDA Image Processing, PRIA 2010 Proceedings, 2010, pp. 176-179.
10. NVIDIA CUDA C Best Practices Guide, Santa Clara, 2010, 75p.