УДК 621.397

ПРОГРАММИРОВАНИЕ ЗАДАЧ ВОССТАНОВЛЕНИЯ ИСКАЖЕННЫХ ИЗОБРАЖЕНИЙ НА С/С++ В СИГНАЛЬНЫХ МИКРОПРОЦЕССОРАХ ФИРМЫ TEXAS INSTRUMENTS

К.А. Кирьянов, В.С. Сизиков

Рассматривается инструментальная реализация алгоритмов восстановления искаженных (смазанных, дефокусированных и (или) зашумленных) изображений. Рассмотрены особенности программирования на языке С/С++ ранее разработанных и опубликованных алгоритмов для прямой, а также более сложной и важной для практики обратной задачи.

Ключевые слова: искажение изображения, прямая и обратная задачи, язык С/С++, сигнальный микропроцессор, персональный компьютер.

Введение. Реализуемые методы и алгоритмы

Известные задачи восстановления (реконструкции, реставрации) искаженных (смазанных, дефокусированных и (или) зашумленных) изображений сформулированы в работах [1–12] и др. Они обычно описываются набором одномерных интегральных уравнений (ИУ) Фредгольма I рода типа свертки,

$$Aw \equiv \int_{-\infty}^{\infty} h(x-\xi)\, w_y(\xi)\, d\xi = g_y(x) + \delta g\ ,\quad a \le x \le b\ ,\quad c \le y \le d\ , \tag{1}$$

или одним двумерным ИУ Фредгольма I рода типа свертки,

$$Aw \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x-\xi, y-\eta)\, w(\xi,\eta)\, d\xi\, d\eta = g(x,y) + \delta g\ ,\quad a \le x \le b\ ,\quad c \le y \le d\ , \tag{2}$$

где h – функция рассеяния точки (ФРТ), в большинстве случаев пространственно-инвариантная (разностная); A – оператор; w и g – распределение интенсивности по истинному и искаженному изображениям соответственно; δg – помеха. В (1) ось x направлена вдоль смаза, а y играет роль параметра. Набор уравнений (1) часто используется в задаче смазывания, а (2) – в задаче дефокусирования изображения [2–11].

Задача решения ИУ (1) и (2) является некорректной (существенно неустойчивой), поэтому ИУ (1) при каждом y часто решают методом квадратур или методом одномерного преобразования Фурье (ПФ) с регуляризацией Тихонова [7, 9–11], а ИУ (2) – методом двумерного ПФ с регуляризацией Тихонова [2–7, 10, 11]. Для решения ИУ (1) и (2) используют также метод параметрической фильтрации Винера [4, 5, 7, 9–11], алгоритм Люси–Ричардсона [5] и метод «слепой» деконволюции [5]. Для решения ИУ (1) или (2) можно использовать также метод итеративной регуляризации Фридмана [10, 11, 13] (о других методах см. [4, 5, 13]). Эти методы могут быть реализованы не только на универсальных ЭВМ или персональных компьютерах (ПК) [3, 5, 10, 11], но и на спецпроцессорах, в частности, сигнальных микропроцессорах или систолических процессорах [8, 12, 14–16]. Использование спецпроцессоров ведет к уменьшению габаритов обрабатывающих устройств и позволяет повысить качество прибора наблюдения (фотоаппарата, телескопа, микроскопа, спектрометра и т.д.) за счет встроенных алгоритмов обработки изображения.

В работе рассматриваются особенности задачи реализации методов и алгоритмов решения ИУ (1) и (2) применительно к восстановлению искаженных изображений с помощью цифровых сигнальных процессоров (ЦСП) и программируемых логических интегральных схем (ПЛИС) [8, 12, 14–16]. В работе [12] уже рассмотрена специфика реализации на спецпроцессорах метода квадратур и метода ПФ с регуляризацией Тихонова применительно к ИУ (1). В настоящей работе остановимся на методе итеративной регуляризации Фридмана решения ИУ (2).

Метод итеративной регуляризации Фридмана

Полагаем, что ядро h уравнения (2) симметрично, т.е. $h(x,y) = h(|x|,|y|)$. Тогда метод Фридмана для решения ИУ (2) описывается следующим соотношением [10, 11, 13]:

$$w_k(x,y) = w_{k-1}(x,y) + \nu\left[g(x,y) - \int_a^b\int_c^d h(x-\xi, y-\eta)\, w_{k-1}(\xi,\eta)\, d\xi\, d\eta \right], \tag{3}$$

где $w_{k-1}(x,y)$ и $w_k(x,y)$ – решения, полученные в $(k-1)$-й и k-й итерациях (приближениях) соответственно, при этом $w_0(x,y)$ – начальное приближение; $k = 1,2,3,\ldots$ – номер итерации; ν – параметр, удовлетворяющий условию $0 \le \nu \le 2/\|A\|$, где $\|A\| = \left\{ \int_a^b\int_c^d h^2(x,y)\, dx\, dy \right\}^{1/2}$ – норма ФРТ.

Для задач дефокусирования $w_k(x,y)$ – изображение, получаемое в k-й итерации, а $g(x,y)$ – правая часть уравнения (искаженное изображение). Метод Фридмана весьма чувствителен к начальному при-

ближению, которое при реализации обычно полагается равным $w_0(x, y) = 0$. Задавшись каким-то (не обязательно нулевым) начальным приближением и выполняя итерационный процесс (3), в принципе можно получить приближение к искомому решению (истинному изображению). Но, поскольку задача некорректна, а правая часть обычно зашумлена, то до некоторого номера итераций k^* процесс обычно сходится к решению, после чего он начинает расходиться [10, рис. 2.70; 13, рис. 21]. Поэтому нужно ввести критерий останова процесса, т.е. выбора числа итераций k^*, которое играет роль параметра регуляризации [13, с. 272–275].

Особенности программной реализации алгоритмов на PC и DSP

Оценим объем памяти, требуемый для размещения одного изображения. Для примера рассмотрим серое изображение размером 512×512 = 262 144 пикселей, записываемое с разрядностью 8 бит на пиксель. Очевидно, что для хранения всего изображения необходимо приблизительно 2 Мбит памяти. Для реализации метода Фридмана требуется хранить 3 изображения: g, w_{k-1} и w_k, что составляет 6 Мбит памяти. Эта величина не критична для ПК, но сопоставима с общим объемом памяти современных высокопроизводительных ЦСП. Таким образом, ключевым моментом рассматриваемой задачи для ЦСП является оптимизация объема используемой оперативной памяти.

Реализация алгоритмов на ПК. На ПК в системе Windows на одну точку изображения отводится 32 бита (4 Б). Из них первый байт (A) отвечает за прозрачность изображения, а три других байта (R, G и B) содержат значения интенсивностей по соответствующим цветовым каналам ($R_{\text{исх}}$, $G_{\text{исх}}$, $B_{\text{исх}}$). В данной работе мы рассматриваем обработку серых изображений, поэтому полагаем по умолчанию $A = 255$ и $R = G = B = (R_{\text{исх}} + G_{\text{исх}} + B_{\text{исх}})/3$. Данные о каждой точке имеют тип BYTE (или unsigned char), значения которого лежат в пределах 0–255, в то время как при решении прямой и обратной задач нужны данные типа double. Поэтому при программной реализации алгоритмов требуется преобразовывать исходные данные в формат double.

При решении прямой задачи (моделировании искажений изображения) часто используются искусственные граничные условия [4, 5], в частности, дополнение исходного изображения нулями вне границ. Однако можно использовать более естественный способ – усечение изображения [7, 9–11]. Кроме того, для уменьшения эффекта Гиббса при решении обратной задачи (задачи восстановления изображения) можно использовать размытие краев изображения [7, 9–11]. Если восстановление осуществляется с помощью алгоритма быстрого преобразования Фурье (БПФ), что особенно характерно для задачи дефокусирования (решения уравнения (2)), то может потребоваться дополнение исходного изображения нулями до размера $2^M \times 2^N$. Чтобы данные не выходили за значение 255, вводится нормировка ФРТ, после чего вычисляется свертка с нормированной ФРТ (прямая задача). Для вывода на экран полученного искаженного изображения осуществляется обратное преобразование типа double в тип BYTE. Дробная часть данных при этом теряется, поэтому числовой массив типа double необходимо сохранять для решения обратной задачи.

а б

Рис. 1. Дефокусированное и зашумленное изображение с размытием краев (а); изображение, восстановленное методом итеративной регуляризации Фридмана ($\nu = 1/\| A \|$, число итераций $k^* = 200$) (б)

Перейдем к анализу обратных задач. В обратной задаче смазывания решается уравнение (1) методом квадратур или ПФ с регуляризацией Тихонова, а также методом параметрической фильтрации Винера, итераций Фридмана и т.д. Особенности методов изложены в работах [7–12]. В обратной задаче дефокусирования, в случае пространственно-инвариантной ФРТ [2–12] решается уравнение (2) методом двумерного ПФ с регуляризацией Тихонова (или другим методом). В конце восстановления изображения путем вычисления обратного ПФ от регуляризованного спектра [7–12] из результата берется только действительная часть, а мнимая часть обратного ПФ отбрасывается. Если же ФРТ является пространствен-

но-зависимой, то решение целесообразно осуществить рассмотренным выше методом итеративной регуляризации Фридмана [10–13]. После восстановления изображения необходимо снова преобразовать данные в формат BYTE и отобразить восстановленное изображение. Пример восстановления дефокусированного (и зашумленного) изображения методом Фридмана приведен на рис. 1.

Реализация алгоритмов на ЦСП. Большинство задач цифровой обработки основано на численных методах работы с матрицами, а также на методах реализации свертки и дискретных ортогональных преобразований [15–20]. Как отмечалось в [12], основная сложность в программировании алгоритмов с помощью ЦСП – это чрезвычайно ограниченные размеры оперативной памяти. В работах [8, 12] был обоснован выбор элементной базы для решения рассматриваемых задач, в том числе, микропроцессора TMS320C6457, который имеет не более 2 МБ оперативной памяти. Как отмечалось в [12], для методов, основанных на дискретных ортогональных преобразованиях, в частности, на ПФ, одним из оптимальных по быстродействию и по распределению памяти является БПФ. Для двумерного БПФ при получении двумерного Фурье-спектра из исходного изображения нужен исходный двумерный массив (изображение) и еще два одномерных массива для временных данных по столбцам. В C/C++ двумерный массив – это длинный одномерный массив с расположением элементов по строкам. В связи с этим для вычисления БПФ по столбцам необходимо брать элемент текущего столбца и записывать его во временный массив, длина которого равна степени 2.

ЦСП TMS320C6457 – это процессор с фиксированной точкой, в то время как необходимо работать с данными типа double и выполнять операции с плавающей точкой. Как отмечено в [12], ядро микропроцессора имеет 32 разряда, а данные типа double имеют 64 разряда. Для преодоления этого противоречия фирмой Texas Instruments [21] разработана библиотека IQmath [22] для работы с виртуальной точкой, с помощью которой ЦСП с фиксированной точкой (процессоры с ядром C64x+) реализуют арифметику с плавающей точкой. Процессоры с плавающей точкой (C67x+) работают на более низких частотах. Если процессор C6457 работает на частоте 1 ГГц, то тактовая частота любого из процессоров C67x+ не превышает 400 МГц. Также ощутима разница в размерах оперативной памяти: если у процессора C6457 ее размер составляет 2 МБ (кэш L2 внутри кристалла), то у большинства процессоров серии C67x+ он не превышает 300 КБ.

Во всех микропроцессорах старший разряд – всегда знаковый. На рис. 2 показано перемножение двух чисел с точкой. Для отображения результатов перемножения двух четырехразрядных чисел с полной точностью требуется 8 разрядов. Но точка перемещается также в результате перемножения. Для обеспечения необходимой точности достаточно правильно извлечь 4 разряда, как это показано на рис. 2.

Множимое и множитель имеют по 4 разряда: один до точки и три – после нее. То же самое относится и к результату. Следовательно, процессор должен иметь возможность работать с данными типа double, которые имеют 64 разряда, несмотря на то, что ядро процессора имеет только 32 разряда. Микропроцессоры серии C64x+ способны выполнять за 1 такт две операции умножения. С помощью библиотеки IQMath можно реализовать программу так, чтобы извлекать только нужные разряды из результата. Заметим, что в принципе это можно сделать и без использования упомянутой библиотеки, но тогда код становится менее читаемым и более громоздким.

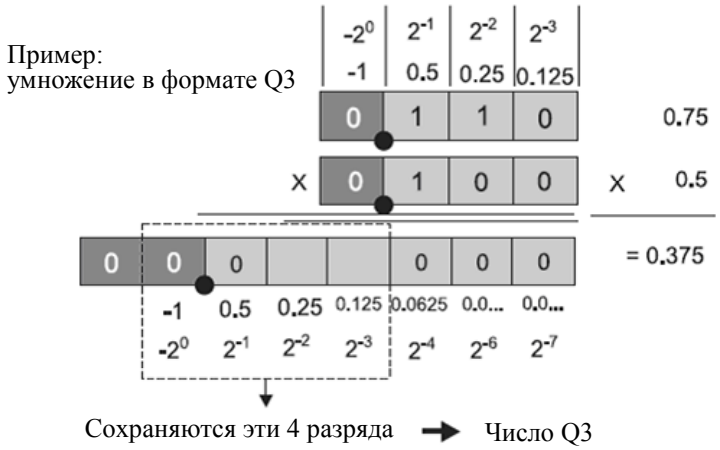

Рис. 2. Перемножение двух чисел с точкой

На рис. 3 показаны три примера кода: первый записан для операций с плавающей точкой, второй реализован только на языке C, третий (Q-вариант) реализован с помощью библиотеки IQMath. Следует заметить, что 32-разрядные данные для DSP C64x+ представлены типом данных int, а 64-разрядные данные – типом long long. После умножения происходит сдвиг на Q разрядов, после чего следует операция суммирования (накопления). Рисунок наглядно показывает, насколько проще и нагляднее код с использованием библиотеки IQMath по сравнению с кодом при реализации только на языке C для ЦСП.

Для выполнения свертки и различных ортогональных преобразований для микропроцессоров TI DSP C64x+ существует библиотека dsplib, в которой имеются написанные библиотеки функций БПФ, оптимизированные библиотеки векторно-матричных перемножений и различных операций с матрицами. Это существенно упрощает работу по написанию кода по сравнению с ситуацией двадцатилетней давности, когда еще не существовали С-компиляторы для ЦСП и разработчики были вынуждены работать исключительно на языке Ассемблер [23].

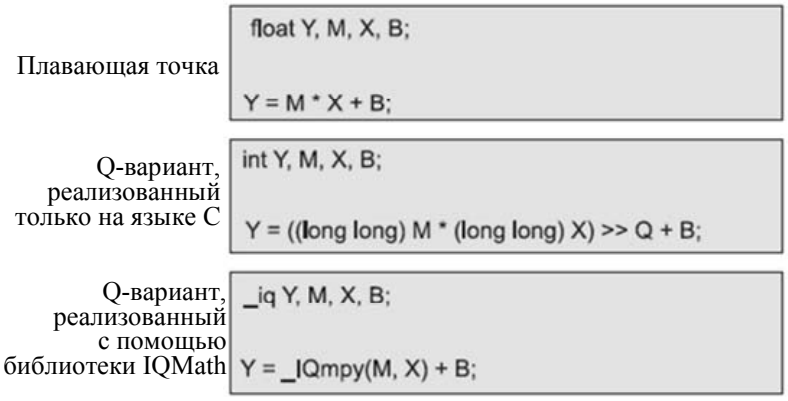

Рис. 3. Три примера кода

Отметим, что производителем Einfochips Limited поставляется отладочный модуль TMDSEVM6457L на базе микропроцессора TMS320C6457 фирмы Texas Instruments. Кроме памяти, расположенной непосредственно внутри кристалла микропроцессора, на борту модуля имеется две микросхемы памяти общим размером 256 МБ и JTAG-интерфейс для пошаговой отладки программ, а также HPI-порт для обмена данными, который может служить для подключения оценочных модулей АЦП и ЦАП через плату сопряжения.

Заключение

В работе рассмотрены сравнительные особенности инструментальной реализации методов и алгоритмов восстановления искаженных (смазанных, дефокусированных, зашумленных) изображений на персональных ЭВМ (PC) и цифровых сигнальных процессорах (DSP). Анализ выполнен применительно к решению прямой и обратной задач смазывания и дефокусирования изображений, описываемых одномерными и двумерными интегральными уравнениями, решение которых осуществляется устойчивыми методами (в работе основное внимание уделено методу итеративной регуляризации Фридмана).

Особенности программирования изложены применительно к языку C/C++. Установлено, что наиболее быстрая реализация алгоритмов достигается в случае разработки кода на языке C/C++ с использованием специализированных библиотек фирмы Texas Instruments под каждое ядро процессора. Для создания быстродействующих алгоритмов необходимо использовать Ассемблер, который позволяет наиболее полно адаптировать код к типу процессора и периферийной схемотехники. Однако в этом случае время написания кода и трудоемкость работы велики. На практике для решения задач чаще всего используется код, написанный на языке C/C++, в некоторых случаях с использованием подпрограмм на языке Ассемблер.

Конвейерная архитектура, реализованная в процессорах C64x+, позволяет выполнять несколько операций одновременно, но требования к алгоритму при этом таковы, что данные выполняемых операций не должны зависеть друг от друга не только в момент выполнения операций, но и для следующего набора команд в конвейер. В этой связи алгоритмы для DSP необходимо реализовывать таким образом, чтобы его оптимизация на уровне конвейера производилась наиболее эффективным образом.

Выявлено, что в среде Code Composer Studio v4.2.0 имеется хороший инструментарий для Фурье-анализа входных последовательностей, а также хорошая графическая оболочка для построения и анализа временных зависимостей, что существенно упрощает работу с необходимым математическим аппаратом.

Проанализированы возможности ПК и ЦСП по быстродействию и памяти, а также по разрядности и форматам чисел. В качестве элементной базы выбран микропроцессор TMS320C6457 – процессор с фиксированной точкой, а вычисления, связанные с обработкой изображений, нужно вести с плавающей точкой. Показано, что для перехода к плавающей (виртуальной) точке целесообразно использовать разработанную фирмой Texas Instruments библиотеку IQmath.

Литература

1. Бейтс Р., Мак-Доннелл М. Восстановление и реконструкция изображений. – М.: Мир, 1989. – 336 с.
2. Бакушинский А.Б., Гончарский А.В. Некорректные задачи. Численные методы и приложения. – М.: Изд-во МГУ, 1989. – 199 с.

3. Сизиков В.С. Математические методы обработки результатов измерений. – СПб: Политехника, 2001. – 240 с.
4. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2006. – 1072 с.
5. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB. – М.: Техносфера, 2006. – 616 с.
6. Воскобойников Ю.Е. Комбинированный нелинейный алгоритм восстановления контрастных изображения при неточно заданной аппаратной функции // Автометрия. – 2007. – Т. 43. – № 6. – С. 3–18.
7. Сизиков В.С., Римских М.В., Мирджамолов Р.К. Реконструкция смазанных и зашумленных изображений без использования граничных условий // Оптический журнал. – 2009. – Т. 76. – № 5. – С. 38–46.
8. Кирьянов К.А. Инструментальная реализация алгоритмов реконструкции искаженных изображений // Труды 20-й Междунар. конф. «GraphiCon–2010». – СПб: СПбГУ ИТМО, 2010. – С. 188–191.
9. Сизиков В.С. Прием «усечение–размытие–поворот» для восстановления искаженных изображений // Оптический журнал. – 2011. – Т. 78. – № 5. – С. 18–26.
10. Сизиков В.С. Обратные прикладные задачи и MatLab. – СПб: Лань, 2011. – 256 с.
11. Сизиков В.С. Интегральные уравнения и MatLab в задачах томографии, иконики и спектроскопии. – Saarbrücken : LAP (LAMBERT Academic Publishing), 2011. – 252 с.
12. Кирьянов К.А., Сизиков В.С. Применение сигнальных микропроцессоров в задачах реконструкции искаженных изображений // Изв. вузов. Приборостроение. – 2011. – Т. 54. – № 7. – С. 20–26.
13. Верлань А.Ф., Сизиков В.С. Интегральные уравнения: методы, алгоритмы, программы. – Киев: Наук. думка, 1986. – 544 с.
14. Кухарев Г.А., Тропченко А.Ю., Шмерко В.П. Систолические процессоры для обработки сигналов. – Минск: Беларусь, 1988. – 127 с.
15. Кухарев Г.А., Тропченко А.Ю. Систолический процессор для обращения матриц // Изв. вузов. Приборостроение. – 1990. – Т. 33. – № 11. – С. 23–27.
16. Тропченко А.Ю. Аппаратные средства для цифровой обработки сигналов: Уч.-методич. пособие по дисциплине «Методы обработки сигналов и изображений». – СПб: СПбГУ ИТМО, 2005. – 138 с.
17. Рабинер Л., Гоулд В. Теория и применение цифровой обработки сигналов. – М.: Мир, 1978. – 848 с.
18. Царев А.П. Алгоритмические модели и структуры высокопроизводительных процессоров цифровой обработки сигналов. – Szczecin: Informa, 2000. – 237 с.
19. Поршнев С.В. Вычислительная математика. Курс лекций. – СПб: БХВ-Петербург, 2004. – 304 с.
20. Киреев В.И., Пантелеев А.В. Численные методы в примерах и задачах. – М.: Высшая школа, 2006. – 480 с.
21. Texas Instruments. Digital Signal Processors [Электронный ресурс]. – Режим доступа: http://www.ti.com/dsp, свободный. Яз. англ. (дата обращения 18.10.2012).
22. Фингер Р. Библиотека функций IQMath для C64x+ с плавающей точкой в ЦСП с фиксированной точкой. Компоненты TI // Бюллетень научно-технич. информации. – 2011. – Вып. 2(30).
23. Blonstein S., Katorgi M. eXpressDSP For Dummies (eXpressDSP для «чайников»): Пер. с англ. – Новосибирск: 2004. – 101 с.

Кирьянов Константин Александрович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, kiryancon@front.ru

Сизиков Валерий Сергеевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, доктор технических наук, профессор, sizikov2000@mail.ru