УДК 004.93'1; 004.932

А.И. Кравчонок

АЛГОРИТМЫ МЕДИАННОЙ ФИЛЬТРАЦИИ С ОКНОМ 3×3 НА ОСНОВЕ НЕПОЛНОЙ СОРТИРОВКИ ПРЯМЫМ ВЫБОРОМ

Предлагаются быстрые алгоритмы вычисления медианного фильтра с окном 3×3 на персональном компьютере. Новые алгоритмы медианной фильтрации изображений с помощью двойной и четверичной неполной сортировки прямым выбором элементов окна фильтра позволяют ускорить выполнение медианной фильтрации изображений и применять их в режиме реального времени в программах обработки изображений для окна фильтра 3×3.

Введение

Медианный фильтр широко применяется для обработки изображений и считается одним из лучших фильтров, так как обладает хорошим сглаживающим действием без существенного размытия границ. Однако из-за своей нелинейности медианные фильтры работают значительно медленнее, чем линейные, и поэтому их редко применяют, например, для обработки видеопоследовательностей изображений в режиме реального времени на персональных компьютерах. Из-за недостаточного быстродействия алгоритмов в задачах, требующих высокой скорости обработки изображений, зачастую медианную фильтрацию используют в виде аппаратных реализаций либо на многопроцессорных системах в параллельном режиме. В различных приложениях анализа и обработки видеопоследовательностей в режиме реального времени быстрый алгоритм медианной фильтрации мог бы улучшить качество обработки изображений [1].

В задачах обработки видеопоследовательностей изображения часто имеют небольшой размер (порядка 720×576), и при их медианной фильтрации целесообразно использовать небольшие окна фильтра – 2×2, 3×3, 5×5. Чем больше окна фильтра, тем более крупные детали исчезают с изображения, а само оно размывается, что ухудшает его качество. Наиболее приемлемым для таких задач является окно медианного фильтра 3×3. Таким образом, целесообразно построить быстрый алгоритм медианной фильтрации для медианного фильтра именно с таким размером окна.

Задачи поиска медианы и сортировки являются тесно связанными между собой, поскольку зачастую поиск медианы в окне фильтра ведется при помощи одного из алгоритмов сортировки. Существуют различные алгоритмы, повышающие быстродействие медианной фильтрации. Подходы, реализуемые для ускорения работы медианного фильтра (за исключением аппаратной и параллельной реализации), можно условно разделить на следующие направления:

1) применение неполной сортировки элементов окна для поиска медианы (алгоритм поиска медианы при помощи неполной быстрой сортировки) [1–3];

2) учет данных, полученных на предыдущих шагах алгоритма (алгоритм быстрой медианной фильтрации Хуанга и его модификации, алгоритм учета предыдущих разностей Мушкаева, алгоритм медианной фильтрации слиянием упорядоченных столбцов) [4–9];

3) реализация алгоритма без применения условных операторов (для алгоритмов, позволяющих это делать) [10–14].

В идеале быстрый алгоритм медианной фильтрации должен удовлетворять пп. 1–3, однако на практике в различных быстрых алгоритмах учитываются не все возможные усовершенствования алгоритма медианной фильтрации, так как реализация одного из подходов может сделать реализацию другого невозможной. Например, алгоритм быстрой медианной фильтрации Хуанга позволяет учитывать и использовать на каждом шаге информацию, накопленную за предыдущие шаги, однако его трудно реализовать без применения условных операторов, так же, как и алгоритм медианной фильтрации на основе неполной быстрой сортировки [1, 3, 4, 15, 16].

1. Реализация программ медианной фильтрации без применения операторов условных переходов

Так как современные процессоры в большинстве своем являются суперконвейерными, быстрее всего они выполняют линейный код без ветвлений. При наличии в программе ветвлений процессор пытается заранее предсказать, какая ветвь программы будет выполняться, однако если предсказание было неверным, то весь конвейер очищается и на его новое заполнение требуется время. Таким образом, ошибки в предсказании ветвлений вызывают значительные задержки в выполнении программы [17]. Очевидно, что невозможно каждый раз угадывать результат выполнения условного оператора, иначе в условном операторе не было бы необходимости. Так как поиск медианы зачастую реализуется посредством сравнения различных элементов окна фильтра, то программа медианной фильтрации включает значительное число операторов условных переходов, что отрицательно сказывается на быстродействии фильтрации. Для ускорения работы фильтра было бы желательно избавиться от операторов условных переходов в программе. Существуют методы замены условных переходов арифметическими операциями, однако применить их удается не в каждом случае, так как некоторые алгоритмы трудно реализовать совсем без использования операций условного перехода [10–14].

Для программной реализации без операторов условных переходов хорошо подходят обменные сортировки, так как операции сравнения двух элементов сортируемой последовательности и возможный их последующий обмен легко реализовать посредством арифметических операций.

Допустим, необходимо найти минимум из значений двух переменных A и B и поместить результат в переменную A. Код на языке С, выполняющий эту процедуру, мог бы выглядеть следующим образом:

$\text{if}\,(A > B)\;\; A = B;$

Зачастую при замене подобного условного перехода арифметическими действиями используют ассемблер [10, 13]. Алгоритм поиска минимума и пересылки его в переменную A заключается в поиске разности R значений переменных A и B и последующем анализе, является ли эта разность отрицательной, посредством проверки специального флага. В случае если разность R является отрицательной, она обнуляется. Затем находится сумма значения переменной A и разности R. В случае если разность $(A - B)$ была отрицательной (т. е. значение переменной A было меньше значения переменной B), значение переменной R равно нулю и, следовательно, значение переменной A после прибавления R не изменится. Если же разность R положительна, т. е. $A > B$, то $A = A + (A - B) = B$. Таким образом, посредством только лишь арифметических операций можно в переменную A поместить минимум из значений A и B. Аналогичным образом можно найти максимум из значений переменных A и B [10, 13].

Несколько усложним задачу. Допустим, необходимо поменять местами значения переменных A и B в случае, если $A > B$. Как известно, чтобы поменять местами значения переменных без использования дополнительной переменной, достаточно найти разность значений переменных, а потом эту разность прибавить к значению второй переменной и отнять от значения первой:

$$R = A - B;$$
$$A = A - R = A - A + B = B;$$
$$B = B + R = B + A - B = A;$$

Чтобы не использовать вспомогательную переменную, разность R можно поместить в одну из переменных A или B, например:

$$A = A - B;$$
$$B = B + A;$$
$$A = A - B;$$

Если добавить к подобной процедуре обмена значений переменных A и B процедуру обнуления разности R в случае, если $A < B$, то получим обмен значений переменных только если $A > B$.

Обнуление разности можно реализовать при помощи ассемблера без применения условных операторов.

В широко применяемой на данный момент библиотеке компьютерного зрения OpenCV [18] реализован аналогичный подход замены условных переходов арифметическими действиями при выполнении медианной фильтрации с окном 3×3, однако для обнуления разности в случае, если $A < B$, в OpenCV используется оригинальный подход, применяющий выборку из таблицы. Заранее формируется таблица T, состоящая из 766 элементов. Значения первых 256 элементов таблицы равны нулю. Значения остальных элементов таблицы рассчитываются по формуле $T_i = i - 256$. Считается, что значения элементов A и B лежат в пределах от 0 до 255. Таким образом, их разность $R = A - B$ находится в пределах $-255 \le R \le 255$. Так как выборку из массива в языке С нельзя вести по отрицательным индексам, то значение R рассчитывается следующим образом:

$$R = T[A - B + 256];$$

Тогда при отрицательной разности $(A - B)$ R примет значение ноль (так как первые 256 элементов таблицы T – ноль), иначе значение R будет соответствовать разности $(A - B)$. Выполнив затем пару арифметических действий: $A = A - R$; $B = B + R$, либо поменяем значения переменных A и B местами, либо оставим их прежними в зависимости от того, меньше A, чем B, или больше.

Подобные реализации алгоритмов без применения условных операторов позволяют значительно ускорить выполнение процедуры медианной фильтрации зачастую в несколько раз, по сравнению с программами, содержащими операторы условного перехода.

Эффективным подходом к реализации медианной фильтрации без применения условных операторов также может являться использование операций MMX (Multimedia Extensions) и арифметики с насыщением [13, 17]. Команды MMX позволяют обрабатывать одновременно несколько элементов окна фильтра и выполнять несколько операций сравнения-обмена параллельно, что значительно повышает быстродействие, а арифметика с насыщением позволяет легко реализовать операцию сравнения-обмена элементов без использования условий.

2. Алгоритм медианной фильтрации на основе неполной сортировки прямым выбором

Сортировка при помощи прямого выбора – обменная сортировка, представляющая собой следующий алгоритм.

Алгоритм 1. Сортировка прямым выбором

1. Выбираем в сортируемой последовательности длиной N минимальный элемент.
2. Меняем местами этот элемент с первым элементом массива.
3. Повторяем процесс для оставшихся $N - 1$ элементов массива до тех пор, пока не останется один последний элемент.

Для выполнения полной сортировки девяти элементов при помощи прямого выбора необходимо $8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 = 36$ операций [2, 3, 15, 16, 19].

Так как необходимо найти лишь медиану последовательности, а не всю отсортированную последовательность, то выполнять алгоритм необходимо только до того момента, пока не будет найдена медиана, т. е. пятый элемент отсортированной последовательности. Таким образом, сортировка выполняется не до конца, а только до получения пятого элемента и поэтому является неполной. Получим алгоритм поиска медианы при помощи неполной сортировки прямым выбором. Количество операций сравнения, необходимых для поиска медианы, будет равно $8 + 7 + 6 + 5 + 4 = 30$.

Алгоритм 2. Неполная сортировка прямым выбором

1. На первом шаге сравниваем первый элемент последовательности с остальными элементами по порядку. Если первый элемент последовательности больше некоторого другого элемента, меняем эти элементы местами. Таким образом, после сравнения первого и всех остальных элементов последовательности на первом месте окажется минимальный элемент последовательности.

2. На втором шаге сравниваем второй элемент последовательности с оставшимися элементами, стоящими правее него (левее рассматриваемого элемента находятся уже отсортированные элементы). Если второй элемент больше некоторого другого элемента, то меняем их местами. После окончания процедуры на втором месте в последовательности будет стоять второй элемент отсортированной последовательности.

3. Продолжаем описанную процедуру сравнения элементов до тех пор, пока не получим пятый элемент массива, который и является медианой.

Алгоритм неполной сортировки можно усовершенствовать, если учесть следующее замечание. Медиана последовательности из девяти элементов является пятым элементом в этой же отсортированной последовательности и имеет следующее свойство: существует ровно четыре элемента, больших медианы, и ровно четыре элемента, меньших медианы. Если существует более четырех элементов последовательности, больших данного, либо более четырех элементов последовательности, меньших данного, то элемент не является медианой. Таким образом, элемент с минимальным значением из любых шести элементов последовательности не может быть медианой, так как существует как минимум пять элементов, больших, чем этот элемент. Следовательно, поиск медианы при помощи описанной выше неполной сортировки прямым выбором можно выполнять следующим образом.

Алгоритм 3. Усовершенствованная неполная сортировка прямым выбором

1. На первом шаге сравниваем первый элемент последовательности со следующими пятью элементами, идущими за ним. Если первый элемент последовательности больше некоторого другого элемента, меняем их местами. Таким образом, после сравнения первого элемента и следующих за ним пяти элементов последовательности на первом месте окажется минимальный из рассматриваемых шести элементов. Как было сказано выше, этот элемент не может быть медианой, также можно утверждать, что он меньше медианы, а следовательно, находится в отсортированном массиве левее медианы.

2. На втором шаге сравниваем второй элемент последовательности со следующими пятью элементами, стоящими правее него (левее рассматриваемого элемента находятся элементы, которые определенно не являются медианой). Если второй элемент больше некоторого другого элемента, то меняем их местами. После окончания процедуры на втором месте в последовательности также будет находиться элемент, который не является медианой и в отсортированном массиве стоит левее нее.

3. Продолжаем описанную процедуру сравнения элементов до тех пор, пока не найдем четыре элемента массива, которые не являются медианой и меньше нее.

4. При помощи описанной выше процедуры ищем среди оставшихся пяти элементов минимальный – этот элемент и является медианой.

Для поиска медианы при помощи описанного выше алгоритма необходимы $5+5+5+5+4=24$ операции сравнения элементов**.**

Заметим, что алгоритм поиска медианы, основанный на сходных принципах, был предложен Паесом [6, 20] и требовал 20 операций сравнения.

3. Алгоритм поиска медианы при помощи двойной неполной сортировки прямым выбором

Дальнейшее совершенствование алгоритма поиска медианы при помощи неполной сортировки прямого выбора основано на использовании данных, полученных на предыдущих шагах работы алгоритма [6].

Рассмотрим два соседних пересекающихся окна 3×3 медианного фильтра (рис. 1). Пусть $C = (c_1, c_2, c_3, c_4, c_5, c_6)$ – последовательность элементов, попадающих как в первое окно фильтра, так и во второе. Элементы последовательности $A = (a_1, a_2, a_3)$ – это элементы, которые попадают только в первое окно фильтра и не попадают во второе, а элементы последовательности $B = (b_1, b_2, b_3)$ – это элементы, попадающие только во второе окно и не попадающие в первое. Таким образом, для поиска медианы первого окна будем производить неполную сортировку последовательности элементов $P_1 = (a_1, a_2, a_3, c_1, c_2, c_3, c_4, c_5, c_6)$, а для поиска медианы второго окна – последовательности $P_2 = (c_1, c_2, c_3, c_4, c_5, c_6, b_1, b_2, b_3)$. Сначала при помощи описанного выше алгоритма неполной сортировки с помощью прямого выбора с небольшой модификацией будем искать медиану второй последовательности P_2.

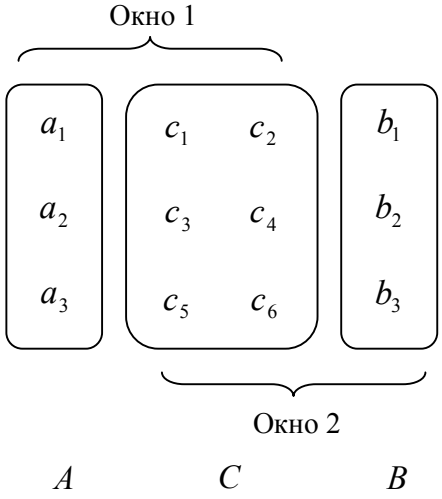

Рис. 1. Два пересекающихся окна фильтра

Обозначим через $S = (s_1, s_2, s_3, s_4, s_5, s_6)$ отсортированную последовательность C. Заметим, что в процессе поиска медианы второго окна выполняются все сравнения элементов, необходимые для получения отсортированной последовательности S. При выполнении алгоритма поиска медианы сначала сравним элемент c_1 с остальными элементами последовательности C: c_2, c_3, c_4, c_5, c_6, при этом находим минимальный элемент последовательности C (первый элемент последовательности S) и сохраняем его в переменную s_1. На следующем шаге сравнивается элемент c_2 с элементами c_3, c_4, c_5, c_6, b_1, однако после выполнения сравнений элемента c_2 с элементами c_3, c_4, c_5, c_6 сохраняем найденный элемент s_2 и лишь потом производим сравнение элемента c_2 с элементом b_1. На последующих шагах, произведя сравнение элемента из последовательности C с остальными элементами этой последовательности, сохраняем результат и после этого производим дальнейшее сравнение с элементами из последовательности B. Таким образом, после выполнения алгоритма получим не только медиану, но и отсортированную последовательность S. Рассмотрим поиск медианы посредством неполной сортировки в предыдущем окне P_1. Выполним сортировку последовательности с конца, при этом будем искать максимальный элемент и помещать его на место последнего элемента. Очевидно, что такой вариант поиска медианы с конца потребует такого же количества операций, что и поиск с начала. Заметим, что при поиске медианы выполняется большое количество операций, связанных с сортировкой элементов последовательности C. В связи с тем что последовательность C уже отсортирована на предыдущем шаге, все операции, связанные с ее сорти-

ровкой, можно не выполнять. Заменяем при выполнении алгоритма элементы множества C соответствующими элементами множества S и выполняем только операции, которые не были выполнены на предыдущем шаге. Таким образом, экономится 15 операций и для поиска медианы первого окна потребуется лишь девять операций.

Последовательность выполнения операций сравнения при поиске двух медиан соседних окон показана на рис. 2, где сравнение двух элементов последовательности и возможный их последующий обмен обозначены дугой.

Таким образом, на поиск двух медиан в соседних окнах требуется $24 + 9 = 33$ операции, следовательно, для поиска одной медианы необходимо выполнить $33/2 = 16{,}5$ операций сравнения и, возможно, обмена элементов.

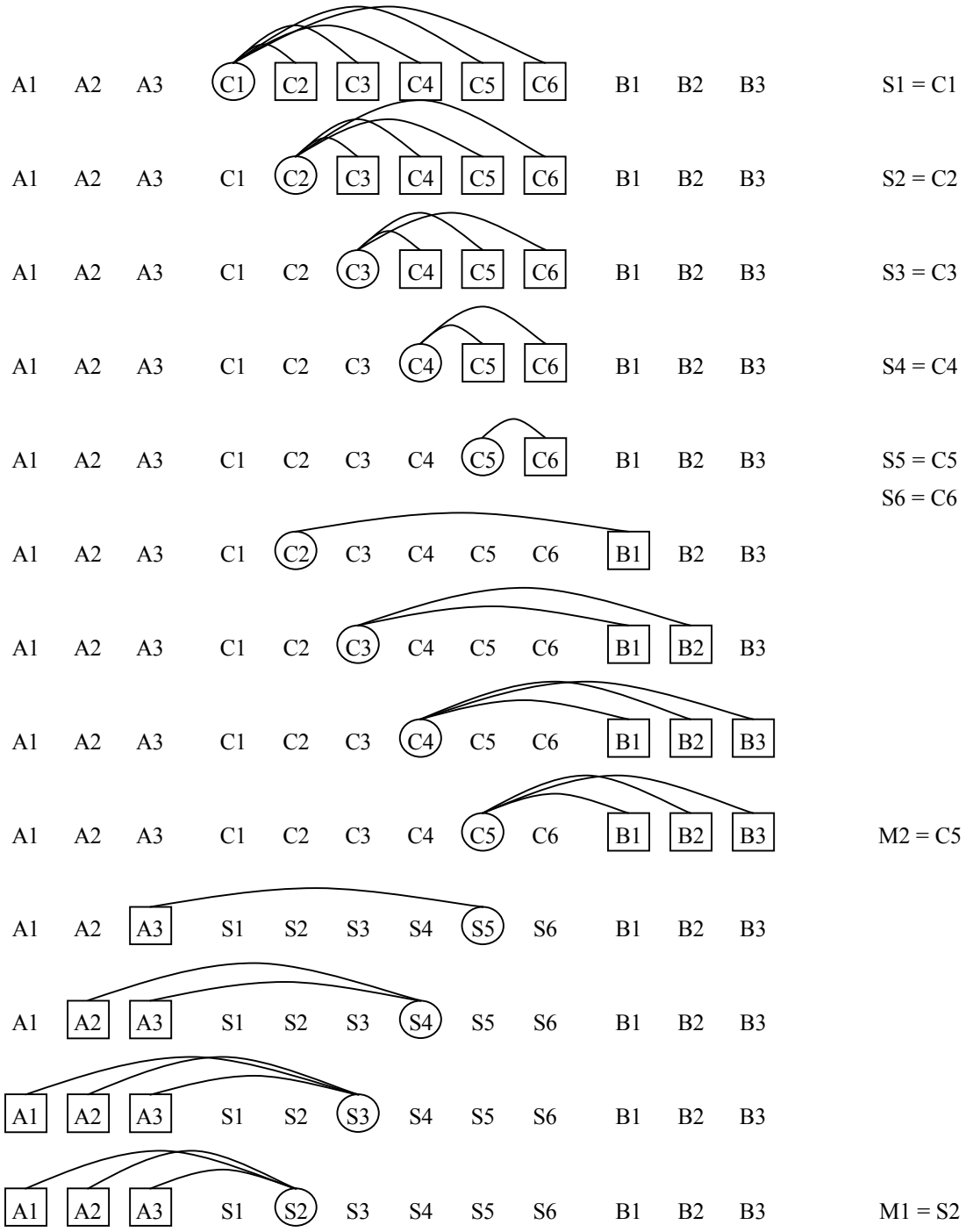

Рис. 2. Порядок сравнения и обмена элементов окон фильтров

4. Алгоритм поиска медианы при помощи четверичной неполной сортировки прямым выбором

В алгоритме неполной двойной сортировки используется тот факт, что соседние окна медианного фильтра пересекаются и выполнять сортировку элементов, попадающих в оба окна дважды, нет необходимости. Заметим, что соседние окна пересекаются не только по горизонтали, но и по вертикали. Будем рассматривать не только соседние по горизонтали окна фильтра, но и пересекающиеся с ними соседние окна по вертикали (рис. 3). На каждом шаге алгоритма будем рассматривать сразу четыре окна и рассчитывать четыре медианы.

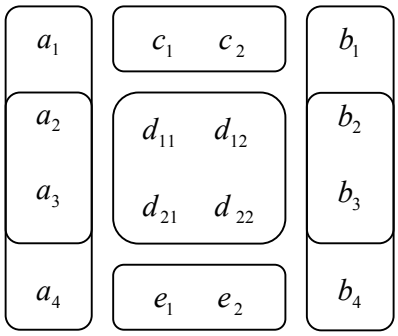

Рис. 3. Четыре соседних пересекающихся по горизонтали и вертикали окна фильтра

Сначала будем сортировать последовательность элементов, попадающих во все четыре окна фильтра $D=(d_{11},d_{12},d_{21},d_{22})$. Затем сортируем две последовательности $P_1 = (d_{11},d_{12},d_{21}, d_{22},c_1,c_2)$ и $P_2=(d_{11},d_{12},d_{21},d_{22},e_1,e_2)$, после чего для каждой последовательности P_1 и P_2 реализуем алгоритм двойной неполной сортировки с соответствующими элементами последовательностей $A_1=(a_1,a_2,a_3)$, $B_1=(b_1,b_2,b_3)$, $A_2=(a_2,a_3,a_4)$, $B_2=(b_2,b_3,b_4)$. При этом получим четыре медианы соответствующих окон фильтра. Для сортировки последовательности D необходимо $3+2+1=6$ операций сравнения. Эти операции выполняются в процессе сортировки последовательности P_1, при сортировке последовательности P_2 выполнять их нет необходимости. Таким образом, для сортировки P_1 необходимо 15 операций сравнения, для сортировки P_2 – всего 9 $(15-6)$ операций. Чтобы получить медианы для каждого окна, необходимо выполнить еще по девять операций сравнения. Всего получим $15+9+4\times 9=60$ операций для поиска четырех медиан. В среднем на одну медиану придется $60/4=15$ операций сравнения, что на 1,5 операции меньше, чем в неполной двойной сортировке.

Заметим, что в реализации медианной фильтрации с окном 3×3 в библиотеке OpenCV используется алгоритм поиска медианы, требующий 19 операций сравнения и обмена элементов окна фильтра.

Сравнительное быстродействие описанных выше алгоритмов медианной фильтрации отображено в таблице.

Таблица

Быстродействие различных алгоритмов медианной фильтрации для полутонового изображения размером 1024×768 ($|d|$ – среднее отклонение медиан соседних окон фильтра в изображении)

Алгоритм поиска медианы	Количество операций сравнения	Быстродействие, мс	
		Реализация с ветвлениями	Реализация без ветвлений
1	2	3	4
Алгоритм 1. Полная сортировка прямым выбором	36	122	39
Алгоритм 2. Неполная сортировка прямым выбором	30	121	38

Продолжение таблицы

1	2	3	4
Алгоритм 3. Усовершенствованная неполная сортировка прямым выбором	24	105	28
Алгоритм 4. Двойная неполная сортировка	16,5	72	20
Алгоритм 5. Четверичная неполная сортировка	15	66	18
Алгоритм медианной фильтрации Хуанга	$7{,}5 + \lvert d \rvert$	71	50
Алгоритм Паеса	20	90	29

Заключение

Проведен анализ возможных путей ускорения работы медианного фильтра на персональном компьютере. На основе анализа предложены новые алгоритмы медианной фильтрации двойной и четверичной неполной сортировкой при помощи прямого выбора элементов окна фильтра. Предложенные алгоритмы для окна фильтра 3×3 превосходят по скорости выполнения известные и позволяют осуществлять медианную фильтрацию изображений в режиме реального времени. Алгоритмы допускают реализацию без применения условных операторов, что значительно повышает их быстродействие.

Список литературы

1. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Дж. Стокман. – М.: БИНОМ. Лаборатория знаний, 2006. – 752 с.

2. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест; пер. с англ. – 2-е изд. – М.: Издательский дом «Вильямс», 2007. – 1296 с.

3. Cormen, T. Introduction to algorithms / T. Cormen, C. Leiserson, R. Rivest. – MIT Press, Cambridge: MA, 2001. – 984 p.

4. Хуанг, Т.С. Быстрые алгоритмы в цифровой обработке изображений / Т.С. Хуанг, Дж.О. Эклунд, Г. Нуссбаумер; под ред. Т.С. Хуанга. – М.: Радио и связь, 1984. – 224 с.

5. Мушкаев, С. Реализация ранжирующих и медианных фильтров на процессоре NM6403 (Л1879ВМ1) / С. Мушкаев // Цифровая обработка сигналов. – 2005. – № 1. – С. 45–47.

6. Kopp, M. Efficient 3×3 Median Filter Computations / M. Kopp // Machine Graphics & Vision. – 1995. – Vol. 4, № 1/2. – P. 79–82.

7. Kravchonok, A. An Algorithm for Median Filtering on the Basis of Merging of Ordered Columns / A. Kravchonok, B. Zalesky, P. Lukashevich // Pattern Recognition and Image Analysis. – 2007. – Vol. 17, № 3. – P. 402–407.

8. Perreault, S. Median Filtering in Constant Time / S. Perreault, P. Hébert // IEEE Transactions on Image Processing. – 2007. – Vol. 16, № 9. – P. 2389–2394.

9. Weiss, B. Fast median and bilateral filtering / B. Weiss // ACM Transactions on Graphics. – 2006. – Vol. 25, № 3. – P. 519–526.

10. Касперски, К. Техника оптимизации программ. Эффективное использование памяти / К. Касперски. – СПб.: БХВ-Петербург, 2003. – 464 с.

11. Магда Ю.С. Аппаратное обеспечение и эффективное программирование / Ю.С. Магда. – СПб.: Питер, 2007. – 352 с.

12. Магда, Ю.С. Ассемблер для процессоров Intel Pentium / Ю.С. Магда. – СПб.: Питер, 2006. – 410 с.

13. Магда, Ю.С. Использование ассемблера для оптимизации программ на C++ / Ю.С. Магда. – СПб.: БХВ-Петербург, 2004. – 496 с.

14. Юров, В.И. Assembler. Практикум / В.И. Юров. – 2-е изд. – СПб.: Питер, 2006. – 399 с.

15. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. – СПб.: Невский Диалект, 2005. – 360 с.

16. Седжвик, Р. Фундаментальные алгоритмы на С. Анализ/Структуры/Сортировка/Поиск / Р. Седжвик; пер. с англ. – СПб.: ООО «Диа Софт ЮП», 2003. – 672 с.

17. Зубков, С.В. Assembler для DOS, Windows и UNIX / С.В. Зубков. – М.: LVR Пресс; СПб.: Питер, 2006. – 608 с.

18. Open Computer Vision Library [Electronic resource]. – Mode of access: http://www.intel.com/technology/computing/opencv. – Date of access: 17.08.2007.

19. Кнут, Д. Искусство программирования. Т. 3. Сортировка и поиск / Д. Кнут; пер. с англ. – 2-е изд. – М.: Издательский дом «Вильямс», 2005. – 824 с.

20. Paeth, A. Median Finding of a 3×3 Grid / A. Paeth, W. Alan // Graphics Gems I. – Academic Press, 1990. – P. 171–175.

Поступила 01.09.07

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: alpha_storm@mail.ru

A.I. Kravchonok

ALGORITHMS FOR MEDIAN FILTERING WITH 3×3 WINDOW ON THE BASIS OF PARTIAL STRAIGHT SELECTION SORT

Fast algorithms of median filter calculation with 3×3 window on a personal computer are proposed. New algorithms of median filtering of images by means of double and quarternary partial straight selection sort of filter window elements allow to accelerate filter performance for real time applications.