ПРОГРАММНАЯ СРЕДА ДЛЯ РАЗРАБОТКИ И ИССЛЕДОВАНИЯ АЛГОРИТМОВ ОЦЕНКИ ДВИЖЕНИЯ ПРИ СЖАТИИ ВИДЕОДАННЫХ

А. М. Кориков, П. В. Потапов
Томский государственный университет систем управления и радиоэлектроники, Россия
e-mail: korikov@asu.tusur.ru, PotapovPavel@mail.ru

In this paper a software environment for algorithms for estimation of motion in data compression is described. Three algorithms for motion estimation were developed, which include full search, diamond search and PMVFast. Results of testing of these algorithms and their characteristic analysis are described.

Введение

Известно, что использование цифровых методов передачи информации приводит к увеличению полосы занимаемых частот и соответственно уменьшению скорости ее передачи. Эта проблема может быть решена путем разработки эффективных методов сжатия (компрессии) компьютерных и телевизионных изображений.

Компрессия — это процесс сжатия данных с целью их представления меньшим количеством бит. Сжатие предполагает наличие пары систем: компрессора и декомпрессора (енкодер, декодер). Компрессор преобразует исходные данные в сжатую форму (занимающую меньшее количество бит) перед передачей или сохранением данных. Декомпрессор преобразует данные из сжатой формы обратно к первоначальному виду.

Пропускная способность компьютерных сетей продолжает расти, высокоскоростное соединение с домашним компьютером стало обычным явлением. Вместительность жестких дисков, флэш-памяти и оптических устройств хранения данных стала больше, чем когда-либо. Стоимость передачи и хранения бит информации становится все меньше и меньше. В связи с этим не очевиден ответ на вопрос: почему прикладывается столько усилий на то, чтобы сделать сжатие более эффективным? Отметим, что сжатие видеоданных имеет два важных преимущества: во-первых, позволяет использовать цифровое видео в таких средах хранения и передачи информации, в которых невозможно использовать видео без компрессии, и, во-вторых, сжатие видеоданных позволяет более эффективно использовать ресурсы среды передачи или хранения информации.

Информация может быть сжата путем устранения из нее избыточности. В системах сжатия без потерь статистическая избыточность удаляется из сигнала таким образом, что он может быть полностью восстановлен получателем без потерь. К сожалению, в

настоящее время методы сжатия без потерь достигают небольшой степени компрессии видео- и аудиоданных. Наиболее часто на практике применяются алгоритмы сжатия видеоданных, основанные на сжатии с потерями. Использование таких алгоритмов позволяет достичь большей степени сжатия, однако недостатком является то, что декодированные данные не полностью идентичны исходным. Цель алгоритма сжатия видеоданных — достижение высокой степени компрессии при минимальных искажениях, вносимых в процессе сжатия.

Большинство видеопоследовательностей содержит высокую временную избыточность, т. е. соседние кадры в последовательности обычно сильно коррелированы между собой (имеют много схожих частей). Разница между соседними кадрами часто соответствует движению в кадре объекта либо движению камеры. Это движение может быть аппроксимировано как линейное движение объекта в кадре либо всего кадра. Таким образом, количество информации, которое должно быть передано, может быть уменьшено в случае, если передавать только разницу между текущим кадром и уже переданным. Это разностное предсказание. Дальнейшее сжатие может быть достигнуто за счет предсказания движения. Каждый блок пикселов в текущем кадре сопоставляется блоку в предыдущем кадре, с которым имеется наибольшее сходство. Смещение между координатами двух блоков называется вектором движения (motion vector). Невязка между текущим блоком и предсказанным блоком кодируется и передается вместе с вектором движения блока. На принимающей стороне оригинальный блок может быть восстановлен путем декодирования невязки значений пикселов блока и добавления ее к значениям блока из предыдущего восстановленного кадра со смещением, соответствующим вектору движения.

Задачей алгоритма оценки движения является поиск такого вектора движения для данного блока, при котором достигается наибольшая степень сжатия. В современных алгоритмах сжатия, использующих компенсацию движения, алгоритмы оценки движения могут занимать до 70 % машинного времени и от их точности зависит эффективность сжатия видеоданных. Именно поэтому актуальна задача разработки новых более быстрых и более эффективных алгоритмов оценки движения. Решение этой задачи затрудняется отсутствием подходящего удобного инструментария. Разработке программного инструмента, позволяющего упростить процесс проверки работоспособности и исследования эффективности алгоритмов оценки движения при сжатии видеоданных, посвящена настоящая работа.

1. Описание программной среды MEFramework

Программная среда MEFramework создана для разработки и исследования методов оценки движения. Состав программной среды представлен на рис. 1. MEFramework реализована в виде DirectShow-фильтра. Такая организация очень удобна, так как позволяет использовать для тестирования видеоданные разных форматов. Необходимо лишь, чтобы в системе стояли DirectShow-кодеки, способные декодировать эти данные.

На вход MEFramework подаются несжатые видеоданные в форматах YV12, YV16 или YUY2. Среда разработки загружает две динамические библиотеки:

— библиотеку алгоритма оценки движения (Motion estimation algorithm dll), которая реализует алгоритм поиска векторов движения в кадре, минимизирующую значение функции стоимости вектора движения;

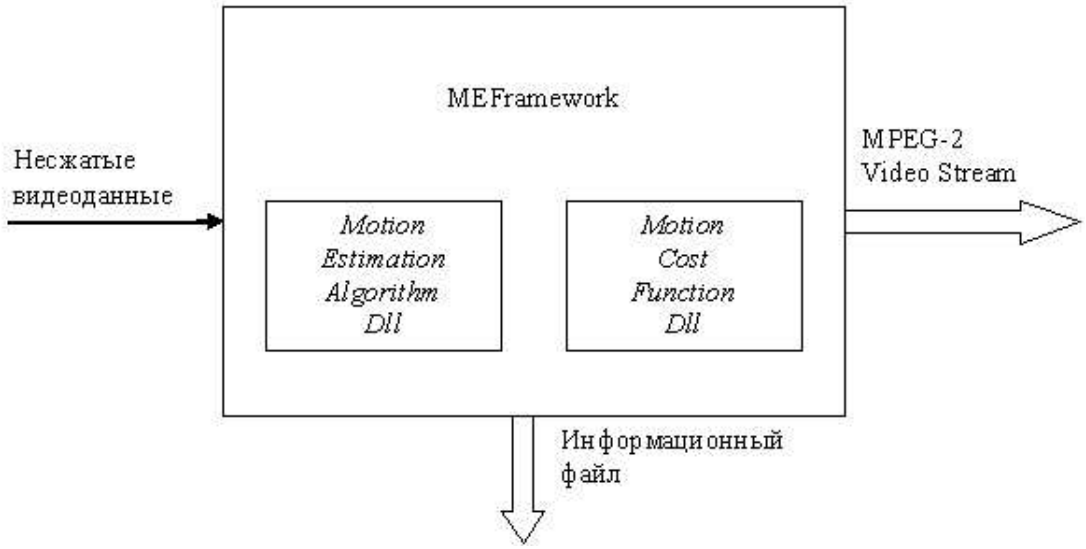


Рис. 1. Состав разработанной программной среды

— библиотеку функции стоимости вектора движения (Motion Cost function dll), которая реализует функцию, позволяющую оценить возможность использования данного вектора движения для компрессии текущего блока.

Для подгружаемых библиотек разработан специальный интерфейс, позволяющий реализовывать алгоритмы оценки движения и функции стоимости векторов движения независимо друг от друга. Преимуществом такой организации программной среды является также то, что пользователю, реализующему подгружаемую библиотеку, можно работать без знания внутреннего строения самой программной среды, ему необходимо лишь соблюдать описанный интерфейс.

Во время работы программной среды происходит сжатие входных видеоданных по стандарту MPEG-2 с использованием исследуемого алгоритма оценки движения. На выходе MEFramework имеем MPEG-2 video elementary stream, который может быть декодирован любым стандартным MPEG-2 декодером. Сжатие происходит с постоянным коэффициентом квантования, поэтому по размеру выходного файла можно сделать вывод об эффективности применяемого алгоритма поиска движения и функции оценки стоимости вектора. Чем меньше размер результирующего файла, тем эффективнее алгоритм оценки движения. Для оценки вычислительной сложности алгоритма оценки движения используется информационный файл со статистикой: при сжатии в этот файл записывается информация о количестве вызовов функции оценки стоимости вектора движения. Таким образом, чем меньше вызовов функции оценки стоимости вектора движения делает алгоритм во время кодирования, тем меньше вычислительных ресурсов он требует.

Для оценки эффективности того или иного алгоритма оценки движения достаточно реализовать его в виде библиотеки с соответствующим интерфейсом и загрузить эту библиотеку в программную среду, затем протестировать на нескольких видеопотоках. В результате получаем две численные оценки, позволяющие сделать вывод об эффективности и вычислительной сложности алгоритма, а также произвести его сравнение с реализованными ранее алгоритмами.

2. Функции оценки стоимости вектора движения

При поиске векторов движения можно использовать различные критерии оптимальности. Выбор критерия является непростой задачей. Требуется подобрать численный критерий, позволяющий оценить, насколько вектор движения подходит для предсказания данного блока. Очевидно, что для оценки оптимальности вектора движения можно провести полную компрессию данного участка кадра, используя оцениваемый вектор, а затем сравнивать размеры блока после сжатия. Обозначим эту меру как $CC\left(v_x, v_y\right)$. Такая мера оценки наиболее точная, так как сравнению подвергаются результирующие размеры сжатого блока, однако полная компрессия требует больших вычислительных ресурсов. В связи с этим на практике данный критерий оптимальности векторов движения используется крайне редко. Наиболее распространены две меры:

$$L1\left(v_x, v_y\right) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left|I_t\left(x+m, y+n\right) - I_{t-1}\left(x+m+v_x, y+n+v_y\right)\right|,$$

$$L2\left(v_x, v_y\right) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left[I_t\left(x+m, y+n\right) - I_{t-1}\left(x+m+v_x, y+n+v_y\right)\right]^2,$$

где v_x и v_y — векторы движения; x, y — координаты текущего блока; I_t — текущий кадр, для которого ведется поиск векторов движения; I_{t-1} — опорный кадр; N — размер блока.

Мера $L1$ с вычислительной точки зрения проще, чем $L2$, так как не использует умножения, однако $L2$ позволяет лучше выявлять резкие различия между текущим блоком и предсказанным. В литературе мера $L1$ часто называется SAD (Sum of Absolute Differences — сумма абсолютных разниц). Более подробно данные меры описаны в [1].

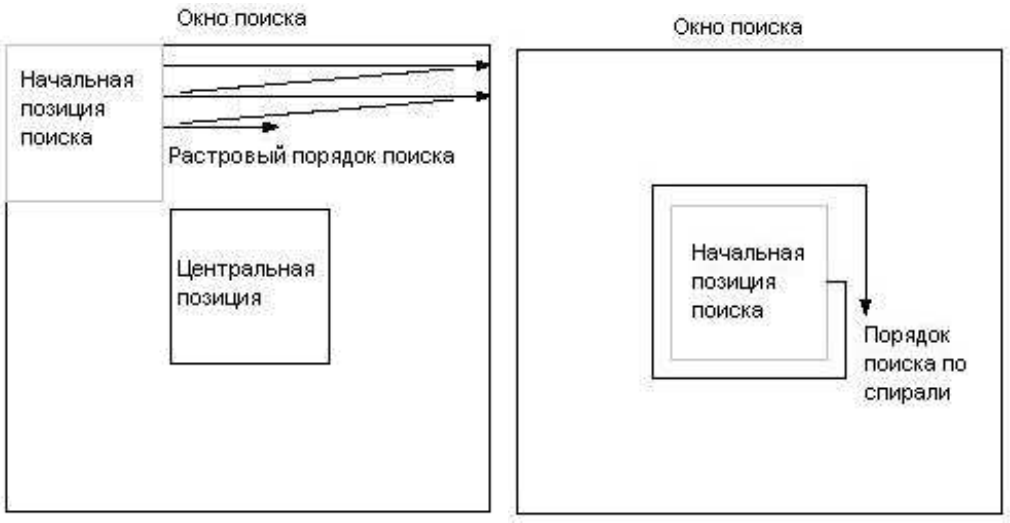
3. Алгоритмы оценки движения при сжатии видеоданных

3.1. Алгоритм полного перебора

Алгоритм полного перебора (Full Search — FS) при оценке движения вычисляет оценочную функцию в каждой точке окна поиска ($\pm S$ позиций относительно позиции $(0, 0)$ — позиции текущего блока). Алгоритм полного перебора гарантирует нахождение минимума оценочной функции в окне поиска, но имеет высокую вычислительную сложность, так как оценочная функция должна быть рассчитана в каждой из $(2S+1)^2$ позиций.

На рис. 2, *а* показан пример стратегии полного перебора. Начальная позиция поиска — в левом верхнем углу окна (координаты $(-S, -S)$), и поиск производится в растровом порядке, пока оценочная функция не будет рассчитана для всех позиций. Обычно в видеопоследовательностях векторы движения концентрируются вокруг позиции $(0, 0)$ и поэтому наиболее вероятно то, что минимум будет найден в окрестности этой позиции.

Таким образом, алгоритм полного поиска упрощается путем перемещения начальной позиции поиска в центр (позицию с координатами $(0, 0)$) и затем продолжением



а *б*

Рис. 2. Полный поиск

вычисления оценочной функции по спирали вокруг этой позиции (рис. 2, *б*). Если используется условие раннего выхода из поиска, то возможно, что вычисление оценочных функций будет прервано до достижения спиралью границы окна поиска (при этом уменьшается вычислительная сложность) [2].

3.2. Алгоритм поиска по алмазу

Алгоритм поиска по алмазу (Fast Diamond search) основан на предположении, что векторы движения в основном центрированы. Алгоритм всегда начинает поиск из центра области поиска путем проверки девяти точек (рис. 3, *а*). Если минимум найден в центре, то проверяются четыре дополнительные точки (рис. 3, *б*) и поиск останавливается. В противном случае предполагаем, что текущий минимум — это центр нового большого алмаза и этот процесс продолжается, пока минимум не окажется в центральной точке (рис. 3, *в* и *г*).

В процедуре оценки совпадения в алгоритме DS текущий блок сравнивается с блоком, расположенным в опорном кадре в соответствии с расположением точек алмаза. Сравнение производится на основе критерия оценки стоимости сжатия, рассчитываемого для всех точек блока. Число точек, в которых происходит вычисление оценочной функции в процессе поиска, непосредственно влияет на вычислительную сложность алгоритма. И если это число может быть уменьшено без влияния на визуальное качество сжатого видео, алгоритм может быть значительно улучшен с точки зрения скорости сходимости.

Блоки с низким (или отсутствующим) движением имеют высокую пространственно-временную корреляцию между собой. В этом случае можно сказать, что существует информационная избыточность между этими блоками. Если использовать эту избыточность подходящим способом, то можно значительно ускорить алгоритм оценки движения. Если между двумя блоками имеется высокая пространственно-временная корреляция, это означает, что соответствующие пиксели в этих блоках имеют одинаковые или близкие значения, что приводит к тому, что мера $L1$ для этих блоков равна или

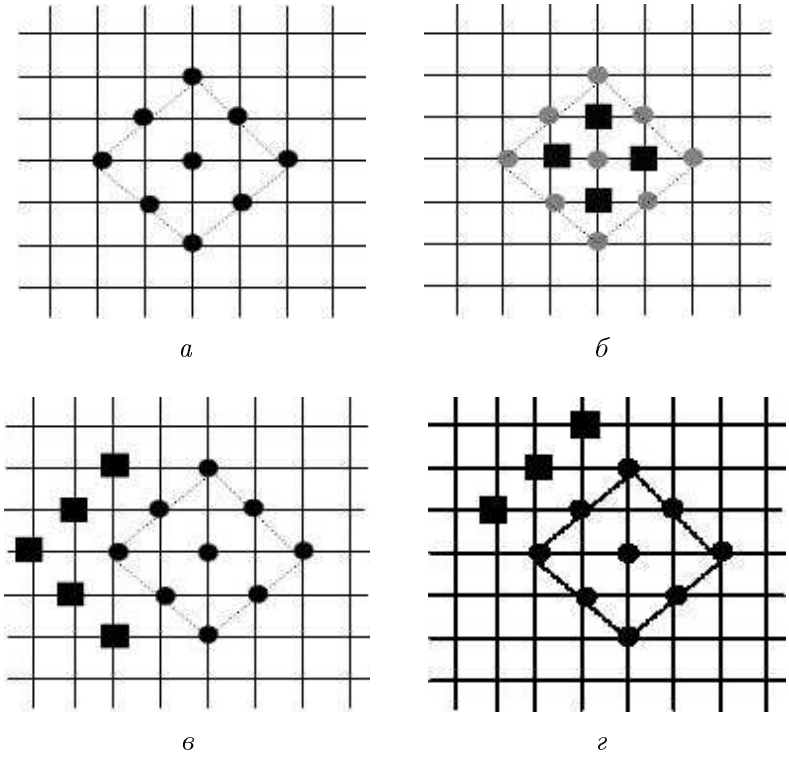


Рис. 3. Поиск по алмазу

близка к нулю. В таком случае вклад всех пикселей блока в вычисление оценочного критерия не обязателен и это увеличивает вычислительную сложность [3].

3.3. Модификация алгоритма поиска по алмазу

Алгоритм PMVFAST (Predictive Motion Vector Field Adaptive Search Technique) является улучшением алгоритма поиска по алмазу. В нем используются те же, что и в алгоритме Fast Diamond search, шаблоны для поиска — малый алмаз и большой алмаз. Однако поиск начинается не в центре поискового окна, как это было в алгоритме поиска по алмазу. В качестве начальной позиции поиска используется один из предсказанных векторов движения (предикторов). Также используются параметры раннего прекращения поиска и усложненное условие выбора шаблона для поиска (большой алмаз или малый алмаз) [4].

В качестве начальных предикторов используются векторы движения трех соседних блоков (левый, верхний и верхний правый), нулевой вектор и медианный предиктор. Медианный предиктор — это вектор движения, который получается путем применения медианного фильтра к значениям векторов движения соседних блоков (левого, верхнего и верхнего правого). Также возможно использование большего количества предикторов, таких как векторы движения других соседних в пространственно-временной области блоков, применение линейных и нелинейных комбинаций этих векторов, например среднее или взвешенное среднее перечисленных выше кандидатов. Выбор наиболее подходящих кандидатов векторов движения называется обобщенным выбором предиктора. Таким образом, векторы левого, верхнего и верхнего правого блоков, их медиана, нулевой вектор и вектор движения соответствующего блока в предыдущем кадре мы

определим как кандидаты набора A. Эти векторы легко использовать, и их нахождение не требует больших вычислительных затрат, к тому же эти векторы сильно коррелированны с вектором движения текущего блока.

В алгоритме PMVFAST используется набор пороговых значений, которые применяются в условиях раннего прекращения поиска. На конкретном этапе поиска текущее минимальное значение оценочной функции сравнивается с пороговым значением и в зависимости от результата поиск прекращается либо немедленно, либо после проверки дополнительных точек или поиск продолжается. Использование фиксированных пороговых значений вызывает ряд проблем, так как пороговое значение может быть слишком велико или слишком мало для некоторых блоков. Выбор порогового значения может зависеть от множества специфических условий сжатия (например, шума, типа движения, разрешения кадра). Таким образом, желательно использовать адаптивное вычисление пороговых значений, так как это может значительно повысить эффективность алгоритма. Предлагается использовать в качестве порогового значения линейную или нелинейную функцию от нескольких доступных параметров, таких как значения оценочной функции для соседних в пространственно-временной области блоков или значения их векторов движения.

В алгоритме PMVFAST сначала вычисляется оценочная функция для медианного предиктора, затем полученное значение сравнивается с порогом T_1, в результате сравнения поиск может быть прекращен после проверки всего одной точки. Порог T_1 адаптивно вычисляется как минимум из значений оценочных функций трех соседних блоков (левого, верхнего и верхнего правого). Нижняя и верхняя границы для этого порогового значения устанавливаются равными 512 и 1024 соответственно.

Существенно улучшена стратегия выбора шаблона для поиска. Эта стратегия позволяет увеличить скорость алгоритма за счет того, что большой алмаз используется реже. На первом этапе вычисляется $\|\mathrm{MedianMV}\|$ — норма медианного предиктора. Если это значение равно нулю и значение оценочной функции для текущего предиктора достаточно велико, то используется большой алмаз, иначе используется малый алмаз.

Для повышения эффективности алгоритма можно воспользоваться тем, что некоторые или даже все векторы движения предикторов одинаковы или близки друг к другу. В этом случае существует большая вероятность того, что искомый вектор движения находится в малой окрестности предикторов. Используя это свойство, можно увеличить скорость алгоритма оценки движения. Например, если векторы всех предикторов одинаковые, то существует бóльшая по сравнению с иной ситуацией уверенность в точности предикторов и можно прервать поиск после N итераций, даже если N мало.

4. Сравнительный анализ алгоритмов

Для тестирования были реализованы три алгоритма оценки движения: алгоритм полного перебора (FS), алгоритм поиска по алмазу (Diamond) и модифицированный алгоритм поиска по алмазу (PMVFAST). Были также реализованы три функции оценки стоимости движения: $L1$, $L2$ и CC.

Результаты тестирования представлены в таблице, где для каждой комбинации алгоритма оценки движения и меры оценки представлены два числа: размер сжатой видеопоследовательности в байтах и количество вызовов оценочной функции в миллионах вызовов.

Результаты тестирования алгоритмов

Алгоритм	Измеряемая величина	Оценочная функция		
		CC	$L1$	$L2$
Diamond	Размер сжатой последовательности, байт	50 269	48 331	47 547
	Количество вызовов оценочной функции, млн	17.462	48.801	43.712
Full search	Размер сжатой последовательности, байт	38 813	46 148	43 067
	Количество вызовов оценочной функции, млн	46 042.514	47 777.402	47 498.389
PMVFAST	Размер сжатой последовательности, байт	43 717	46 992	43 294
	Количество вызовов оценочной функции, млн	12.154	21.259	21.522

Из результатов тестирования алгоритмов видно, что наилучшую степень сжатия обеспечивает алгоритм полного перебора с оценочной функцией CC. Однако при этом требуется большое количество вычислений оценочной функции, к тому же вычислительная сложность меры CC во много раз превышает вычислительную сложность мер $L1$ и $L2$. Меры оценки $L1$ и $L2$ показали схожие результаты, однако функция $L1$ не содержит умножений, что является ее преимуществом при оценке векторов движения.

Наилучший результат по соотношению скорости и степени сжатия показал алгоритм PMVFAST. Степень сжатия этим алгоритмом незначительно меньше по сравнению с алгоритмом полного перебора, но количество вычислений оценочной функции наименьшее. По отношению $L1$ количество вычислений оценочной функции для алгоритма PMVFAST в 2.27 раза меньше по сравнению с алгоритмом Fast Diamond search и в 2247 раз меньше по сравнению с алгоритмом полного перебора. Кроме того, сжатая видеопоследовательность, полученная с использованием алгоритма PMVFAST, на 1.17 % больше видеопоследовательности, полученной с использованием алгоритма полного перебора, и на 2.8 % меньше последовательности, полученной с использованием алгоритма поиска по алмазу.

Таким образом, для достижения наилучшего качества при слабых ограничениях на вычислительные ресурсы рекомендуется использовать алгоритм полного перебора, а для достижения наилучшего соотношения скорости и степени сжатия рекомендуется использовать алгоритм PMVFAST в сочетании с мерой $L1$.

Заключение

Созданная программная среда является удобным инструментарием, позволяющим осуществлять разработку, тестирование и исследование алгоритмов оценки движения при сжатии видеоданных.

В программной среде реализованы три алгоритма оценки движения (алгоритм полного перебора, алгоритм поиска по алмазу и его модификация) и три оценочные функции. Произведено тестирование и сравнение этих алгоритмов. Полученные результаты сравнения соответствуют ожидаемым и согласуются с результатами, полученными другими авторами [2–4].

Список литературы

[1] Richardson E.G. H.264 and MPEG-4 video compression / The Robert Gordon University. Aberdeen, UK. 2003. P. 30–41.

[2] Dzung T.H., Philip M.L., Jeffrey S.V. Efficient cost measures for motion estimation at low bit rates // IEEE Transactions on Circuits and Systems for Video Technology. 1998. Vol. 8, N 4. P. 488–500.

[3] Zhu S., Ma K.K. A new diamond search algorithm for fast block matching motion estimation // Proc. Intern. Conf. Information, Communications and Signal Processing. 1997. Vol. 1. P. 292–298.

[4] Tourapis A.M., Au O.C., Liou M.L. Predictive motion vector field adaptive search technique (PMVFAST) — Enhancing block-based motion estimation // Proc. of Visual Comm. and Image Proc. (VCIP-2001). 2001. Vol. 1. P. 315–325.

Поступила в редакцию 23 августа 2007 г.