УДК 681.3.053

АЛГОРИТМ ПОИСКА ВЕКТОРОВ ПОХОЖЕСТИ ПРИ СЖАТИИ ВИДЕОДАННЫХ

П.В. Потапов, А.М. Кориков*

Отдел проблем информатизации ТНЦ СО РАН, г. Томск
*Томский государственный университет систем управления и радиоэлектроники
E-mail: PotapovPavel@mail.ru, korikov@asu.tusur.ru

Предложен новый алгоритм поиска векторов похожести при сжатии видеоданных. Использование данного алгоритма позволяет улучшить производительность и степень компрессии видео. Проведен сравнительный анализ предложенного алгоритма с аналогами.

Ключевые слова:
Мультимедиа, видеоданные, сжатие, вектор похожести, оценочная функция.

Введение

Пропускная способность компьютерных сетей продолжает расти, высокоскоростное соединение с домашним компьютером стало обычным явлением. Вместительность жёстких дисков, флэш-памяти и оптических устройств хранения данных стала больше чем когда-либо. Стоимость передачи и хранения информации становится всё меньше и меньше, однако проблема сжатия видеоданных не теряет своей актуальности. Возникновение мультимедиа (*multimedia*) как нового научно-технического направления поставило проблему сжатия различных сигналов на первое место. Алгоритмы сжатия видеоданных постоянно совершенствуются, создаются новые стандарты видеокомпрессии. Сжатие видеоданных имеет два важных достоинства. *Во-первых*, сжатие позволяет использовать цифровое видео в таких средах хранения и передачи информации, в которых невозможно использовать видео без компрессии. *Во-вторых*, сжатие видеоданных позволяет более эффективно использовать ресурсы среды передачи или хранения информации [1].

В настоящее время огромное внимание уделяется разработке алгоритмов сжатия видеоданных, использующих компенсацию движения. Так, начиная с 1980 г. организации *ITU* и *MPEG* последовательно выпускают ряд стандартов сжатия, использующие блочную компенсацию движения: *H*.261, *MPEG*-1, *MPEG*-2/*H*.262, *H*.263, *MPEG*-4, *H*.264 (*MPEG*-4 *part*-10) [2]. Использование алгоритмов компенсации движения при сжатии видеоданных позволяет существенно увеличить степень компрессии при том же соотношении сигнал/шум результирующего видеосигнала. Параллельно с разработкой новых стандартов сжатия происходит совершенствование алгоритмов поиска векторов похожести. Алгоритм поиска векторов похожести является важнейшей частью в системе сжатия видеосигнала. От выбора алгоритма поиска векторов похожести напрямую зависит степень компрессии и вычислительная сложность алгоритма сжатия. Поиск векторов похожести является одним из самых ресурсоемких этапов компрессии видеосигнала. Однако сам алгоритм поиска векторов похожести обычно не фиксируется в стандарте сжатия. Таким образом, модуль поиска векторов похожести является одной из немногих частей видеокомпрессора, которую можно свободно подвергать алгоритмической оптимизации. Реализация оптимизированного модуля даёт значительное конкурентное преимущество над другими компрессорами, реализующими тот же стандарт сжатия. Именно поэтому в этой области продолжаются интенсивные исследования, несмотря на то, что уже разработано множество алгоритмов векторов похожести движения для различных задач.

Критерии сравнения алгоритмов поиска векторов похожести

Для сравнения алгоритмов было выбрано два критерия:

1) количество вычислений оценочной функции – N позволяет оценить вычислительную сложность алгоритма поиска векторов похожести;
2) в качестве оценки точности найденных векторов похожести используется размер сжатой видеопоследовательности M, компрессия производится в соответствии со стандартом *MPEG*-2 с постоянным коэффициентом квантования. Для нахождения векторов похожести во время компрессии используются оцениваемый алгоритм.

Оценочная функция

В качестве оценочной функции традиционно используется сумма абсолютных разностей (*SAD* – *sum of absolute differences*). Вычисление данной функции не требует умножений, что делает её привлекательной для поиска векторов похожести. Функция *SAD* для некоторого блока A размером $K \times K$, имеющего координаты (x,y), и вектора похожести V вычисляется следующим образом:

$$SAD(V_x, V_y) =$$

$$= \sum_{m,n=0}^{K-1} \left| I_t(x+m, y+n) - I_{t-1}(x+V_x+m, y+V_y+n) \right|,$$

где I_t, I_{t-1} – значения яркости пикселей текущего и предыдущего кадров [3]; t – номер текущего кадра.

Описание разработанного алгоритма

Для формализации процесса разработки алгоритм поиска векторов похожести был разбит на следующие части:

- поиск начального приближения вектора похожести;
- уточняющий поиск;
- завершающая обработка найденных векторов похожести.

Этап поиска начального приближения вектора похожести очень важен. Так как оценочная функция не является выпуклой и имеет множество оврагов и локальных минимумов, применение метода локальной оптимизации будет не эффективно, если начальная точка поиска выбрана неудачно. Для определения начальной точки поиска применяется алгоритм поиска начального приближения вектора похожести.

Для определения точки начального приближения мы можем использовать найденные векторы похожести соседних блоков в текущем кадре, а также векторы похожести из предыдущего кадра. Использование этой априорной информации и есть основная идея метода поиска начального приближения вектора похожести. Изложим алгоритм реализации этой идеи с помощью рисунка, на котором представлены блоки поиска начального приближения:

1. Векторы соседних блоков MB_1, MB_2, MB_3 (рис. 1) используются для определения так называемого медианного (среднего) вектора [4]. Вычисляем значение оценочной функции для медианного вектора. Сравниваем это значение с пороговой величиной T1, если полученное значение меньше чем $T1$ переходим к уточняющему поиску, в противном случае переходим к следующему шагу 2. Пороговая величина $T1$ вычисляется исходя из значений оценочных функций соседних блоков по формуле:

$$T1 = a_k \min(SAD_1, SAD_2, SAD_3, SAD_4) + b_k ,$$

где a_k, b_k – весовые коэффициенты.

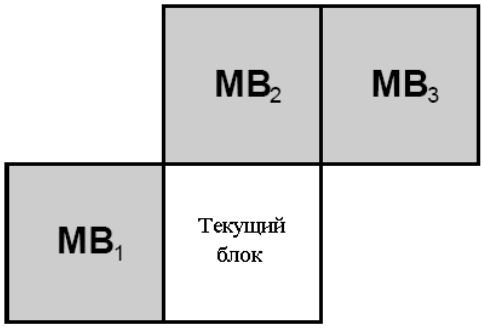

Рисунок. *Блоки, используемые для вычисления медианного вектора*

2. Составляем список векторов-кандидатов. В этот список добавляем найденные векторы соседних блоков в текущем и предыдущем кадрах. Также в список добавляются интерполированные векторы: если в предыдущем кадре вектор движения какого либо из блоков указывал в текущий блок, то этот вектор так же добавляется в список [5].
3. Из списка векторов-кандидатов удаляем близкие вектора. Для всех векторов попарно рассчитывается величина $d(V^1, V^2)=|V_x^1-V_x^2|+|V_y^1-V_y^2|$. Если $d(V^n, V^m)<T2$, то вектор V^m исключается из списка векторов-кандидатов. Экспериментально было получено оптимальное значение параметра $T2$, равное 4 [6].
4. Вычисление оценочной функции производим для всех оставшихся векторов-кандидатов. Вектор-кандидат с наименьшим значением оценочной функции принимается за вектор начального приближения.
5. Если значение оценочной функции в точке начального приближения больше чем $T3=1/2T1$, то принимаем решение о том, что точка начального приближения найдена неудачно и производим её адаптивный поиск методом сканирования в зависимости от размеров поискового окна и разрешения изображения.

В результате работы данного алгоритма получаем точку начального приближения. На следующем этапе происходит уточнение найденного вектора похожести.

Для проведения уточняющего поиска выбран градиентный метод локальной оптимизации [7]. Градиентный метод сводится к следующему итерационном процессу:

1. В точке V^k вычисляется градиент оценочной функции. Вычисление градиента оценочной функции в зависимости от ситуации производится методом прямой (упреждающей), либо обратной (отстающей) конечной разности по формулам:

$$\nabla f(x, y) = \{f(x+1, y) - f(x, y), f(x, y+1) - f(x, y)\},$$
$$\nabla f(x, y) = \{f(x, y) - f(x-1, y), f(x, y) - f(x, y-1)\}.$$

2. Принимаем $V^{k+1} = V^k - \lambda \dfrac{\nabla f(V^k)}{\left\|\nabla f(V^k)\right\|}$, где λ – шаг поиска, вычисляем $f(V^{k+1})$;
3. Если $f(V^{k+1})<f(V^k)$, принимаем V^{k+1} за текущую точку и переходим к шагу 1, иначе уменьшаем λ в два раза и переходим к шагу 2.

Поиск производится до тех пор, пока $\lambda>1$. Результатом поиска является вектор похожести V^k.

Экспериментально доказано, что метод поиска с использованием градиента требует меньше вычислений оценочной функции для нахождения локального минимума, чем методы нулевого порядка, такие как метод Гаусса-Зейделя, метод Хука-Дживса, симплексный метод. Метод локальной оптимизации с использованием градиента требует меньше вычислений оценочной функции по сравнению с методами поиска по шаблону, применяемыми в алгоритмах *Diamond*, *PMVFAST* [5] *EPZS* [3].

После нахождения векторов похожести для всех блоков кадра производится уточняющий обратный проход. На этом этапе проходим все блоки кадра в обратном порядке. По сравнению с первым проходом появляется информация о векторах похожести блоков, расположенных ниже и правее текущего блока. Проверяем значения оценочных функций для этих векторов. Если значение оценочной функции меньше чем для найденного на первом проходе вектора похожести, проводим уточняющий поиск для вектора с наименьшим значением оценочной функции.

Сравнение с аналогами

Проведем сравнение разработанного алгоритма с современными аналогами. В сравнительном анализе участвовали алгоритмы: *CBA* [8], *ADZS* [5], *PMVFAST*, *EPZS*. Алгоритм *EPZS* является одним из наиболее эффективных алгоритмов поиска векторов похожести. Использование этого алгоритма позволяет достичь высокой степени компрессии с при низком количестве вычислений оценочной функции [3].

При сравнении использовался программный продукт – программная среда *MEFramework* [9]. Данная среда была разработана для сравнения алгоритмов поиска векторов похожести при сжатии видеоданных. В ходе сравнения производилось сжатие тестовых видеопоследовательностей в соответствии со стандартом *MPEG*-2 с постоянным коэффициентом квантования с использованием тестируемого алгоритма поиска векторов похожести. В таблицу заносилось количество вычислений оценочной функции *N*, а так же результирующий размер сжатой видеопоследовательности *M*. Количество вызовов оценочных функций для сравниваемых алгоритмов приведено в табл. 1. Значения меры *M*, полученные для исследуемых алгоритмов на тестовых видеопоследовательностях, представлены в табл. 2.

В таблицах предложенный алгоритм обозначен как «*Proposed*». Из табл. 1 видно, что количество вычислений оценочной функции для разработанного алгоритма является наименьшим, по сравнению с аналогами. Из табл. 2 следует, что применение предложенного алгоритма поиска позволяет достичь большей степени компрессии, чем при использовании рассмотренных аналогов.

Таблица 1. *Таблица значений меры N для тестируемых алгоритмов, млн вызовов*

Алгоритм	Тестовая видеопоследовательность			
	1	2	3	4
PMVFAST	21259	4705	7652	5705
CBA	64321	13788	21469	13788
ADZS	47801	8134	12361	9134
NTTS	40000	7134	11000	8134
EPZS	19259	4200	6900	5200
Proposed	14030	3105	5050	3105

Таблица 2. *Таблица значений меры M для тестируемых алгоритмов, Кбайт*

Алгоритм	Тестовая видеопоследовательность			
	1	2	3	4
PMVFAST	48148	11015	14878	12109
CBA	48420	12160	16726	13160
ADZS	48331	12172	18726	13172
NTTS	50332	12200	19041	13230
EPZS	48011	10880	14817	11509
Proposed	46139	10031	14650	10712

Заключение

Разработан и реализован новый алгоритм поиска векторов похожести при сжатии видеоданных. Основным отличием разработанного метода от аналогов является использование градиентного метода локального поиска для уточнения вектора похожести.

Произведен сравнительный анализ разработанного алгоритма с аналогами. По результатам сравнения сделан вывод, что использование предложенного алгоритма позволяет увеличить степень компрессии видеоданных и уменьшить количество вычислений оценочной функции по сравнению с аналогами.

СПИСОК ЛИТЕРАТУРЫ

1. Артюшенко В.М., Шелухин О.И., Афонин М.Ю. Цифровое сжатие видеоинформации и звука. – М.: Дашков и К, 2003. – 426 с.
2. Richardson E.G. H.264 and MPEG-4 Video Compression. – Aberdeen, UK: The Robert Gordon University, 2003. – 306 p.
3. Tourapis A.M. Fast Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation // Proc. of Visual Comm. and Image Proc. (VCIP-2003). – 2003. – V. 2. – P. 1254–1266.
4. ISO/IEC 14496-2: Information technology – Coding of audio-visual objects. – Part 2: Visual, 2001.
5. 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. – V. 1. – P. 315–325.
6. Кубасов Д., Ватолин Д. Обзор методов компенсации движения [Электронный ресурс]. – режим доступа: http://cgm.graphicon.ru/content/view/76/64/. – 13.05.2007.
7. Мицель А.А., Шелестов А.А. Методы оптимизации. – Томск: Изд-во ТУСУР, 2004. – 256 с.
8. Christopoulos V., Cornelis J. A Center-Biased Adaptive Search Algorithm for Block Motion Estimation // IEEE Transactions on Circuits and Systems for Video Technology. – 2000. – V. 10. – № 3. – P. 423–426.
9. Кориков А.М., Потапов П.В. Программная среда для разработки и исследования алгоритмов оценки движения при сжатии видеоданных // Вычислительные технологии. – 2007. – Т. 12. – Спец. вып. 1. – С. 42–50.

Поступила 29.09.2008 г.