Быстрый алгоритм нахождения движения в видеопоследовательностях

С.Ю. Путилин
Московский государственный университет, Москва, Россия
sputilin@graphics.cs.msu.ru

Аннотация

В данной статье представлен алгоритм поблочного нахождения движения в видео. Особенностями этого алгоритма являются добавление векторов движения в набор векторов-кандидатов следующего кадра, поиск областей с новыми объектами и иерархический поиск векторов в этих областях. Предложенный алгоритм сравнивается по скорости и качеству с широко известными алгоритмами и с утилитой MVTools для программы AVISynth.

***Ключевые слова**: нахождение движения, обработка видео, кодирование видео.*

1. ВВЕДЕНИЕ

Даже в кодеках такого старого стандарта, как MPEG-2 [1], используется компенсация движения. Она позволяет устранять избыточность данных во времени для обеспечения высокой степени сжатия. Поскольку от качества компенсации зависит как степень сжатия, так и скорость работы кодека, алгоритмы компенсации движения постоянно совершенствуются. Для ускорения поиска векторов движения были предложены алгоритмы, использующие вектора-кандидаты [2], [4], [5], алгоритмы шаблонного поиска [7] и алгоритмы, использующие иерархическую компенсацию [3].

Рис. 1. Схема алгоритма нахождения движения

В данной статье предлагается алгоритм, который является гибридом вышеописанных подходов, сделанным таким образом, чтобы избавится от недостатков каждого из них. Мы будем использовать иерархический поиск для нахождения движения новых объектов в видео, вектора-кандидаты для объектов, движение которых уже было найдено и шаблоны для уточнения векторов. Тем самым мы избавимся от высокой вычислительной сложности иерархической компенсации, проблемы с определением векторов движения новых объектов у методов с векторами-кандидатами и плохого качества поиска быстрого движения у шаблонных методов.

2. ОПИСАНИЕ АЛГОРИТМА

Алгоритм основан на использовании векторов-кандидатов (ВК) из соседних блоков для текущего обрабатываемого блока. Т.к. поле векторов движения для отдельных объектов или фона характеризуется непрерывностью не только в пространстве, но и во времени, то ВК берутся не только из соседних блоков данного кадра, но и из соседних блоков предыдущего кадра. Схема алгоритма представлена на Рис. 1.

2.1 Расширение набора векторов-кандидатов

В самом начале работы над блоком у нас уже есть некий набор векторов-кандидатов (НВК), который был сформирован во время нахождения движения в предыдущем кадре (формирование этого набора будет рассмотрено в пункте 2.7). В начале работы над текущим блоком (X) происходит расширение НВК за счет векторов движения из соседних блоков, показанных на Рис. 2, т.е. из левого (A), левого верхнего (B), верхнего (C) и правого верхнего (D) блоков.

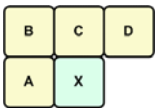

Рис. 2. Расширение НВК

После этого мы переходим к обработке полученного НВК.

2.2 Выбор наилучшего вектора-кандидата

Сначала мы находим максимальную разность векторов в НВК, чтобы определить характер движения в окрестности текущего блока. Пусть у нас в НВК N векторов. Тогда максимальная разность D определяется следующим образом:

$$Dx = \max_{i,j \in [1,N]} \left| x_i - x_j \right|$$
$$Dy = \max_{i,j \in [1,N]} \left| y_i - y_j \right| \qquad (1)$$
$$D = \max\{Dx, Dy\}$$

где x_i, y_i – горизонтальная и вертикальная составляющие i-го вектора из НВК. После того, как вычислена максимальная разность D, мы сравниваем ее с порогом tD и решаем, насколько сложное движение в данной области. Если $D<tD$, то движение в области считается однородным и в качестве наилучшего ВК берется усреднение всех векторов-кандидатов. Если же $D>=tD$, то считается, что движение в области сложное и проводится прореживание НВК. Эту процедуру мы рассмотрим в следующем пункте. После нее

производится вычисление сумм абсолютных разностей (SAD) между текущим блоком и блоками в предыдущем кадре, соответствующими векторам движения из прореженного НВК. ВК с наименьшим SAD считается наилучшим.

2.3 Прореживание НВК

Будем вычислять расстояние между векторами по формуле:

$$Dist = |x_1 - x_2| + |y_1 - y_2| \quad (2)$$

Алгоритм прореживания действует следующим образом:

- Для всех векторов *i* из НВК, начиная с первого, считаются расстояния между ним и каждым из последующих векторов *j* из НВК.
- Если расстояние с каким-либо из последующих векторов *j* будет меньше порога *tD*, то вектор *j* исключается из НВК и в дальнейшем для фильтрации не рассматривается.

Этот процесс направлен на удаление из НВК одинаковых и близких векторов, чтобы уменьшить количество операций по вычислению SAD.

2.4 Допоиск

Сначала допоиск осуществляется с шаблоном, известным в литературе как малый ромб (small diamond pattern или SDP) [4]-[6]. Когда минимум SAD достигнут в средней точке шаблона или если сделано максимальное количество разрешенных шагов, то делается сначала полупиксельное, а затем четвертьпиксельное уточнение вектора движения. Поиск при помощи шаблона проиллюстрирован на Рис. 3. Начало поиска – круг, обозначенный буквой "Ц". Квадратами выделены места, где был достигнут минимум SAD в соответствующих шагах.

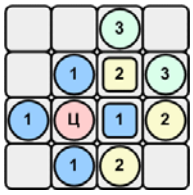

Рис. 3. Допоиск при помощи шаблона "малый ромб"

2.5 Определение правильности вектора

У всех методов компенсации, основанных на использовании векторов-кандидатов, есть общий недостаток: они не сразу находят движение новых объектов. Для того чтобы ускорить этот процесс, автором был предложен метод нахождения блоков, в которых появился новый объект. Будем исходить из предположения, что в таких блоках SAD векторов, найденных при помощи НВК, будет выше, чем в среднем по кадру. Но если брать в качестве порога простое усреднение значений SAD по предыдущему кадру, то из-за шума найдется слишком много блоков, в которых SAD найденных векторов будет выше порога. Если рассматривать модель белого шума с Гауссовским распределением, а SAD векторов как случайную величину, то можно предполагать, что все значения SAD попадут в область $E \pm 3\sigma$, где E – мат. ожидание, σ – среднеквадратичное отклонение. Значения E и 3σ вычисляются для предыдущего кадра. Но просто так их тоже использовать нельзя. Потому что если в видео произойдет смена сцены, то этот порог станет слишком высоким, что нежелательно. Поэтому применяется рекурсивное усреднение этих чисел по всей последовательности по формулам (3).

$$\begin{aligned} E_{av}^t &= k_E E^t + (1 - k_E) E_{av}^{t-1} \\ \sigma_{av}^t &= k_\sigma \sigma^t + (1 - k_\sigma) \sigma_{av}^{t-1} \end{aligned}, \quad (3)$$

где E_{av}^t – усредненное по времени мат. ожидание на кадре t, k_E – коэффициент обновления мат. ожидания, σ_{av}^t – усредненное по времени среднеквадратичное отклонение на кадре t, k_σ - коэффициент обновления среднеквадратичного отклонения. Порог T, с которым сравнивается SAD текущего наилучшего вектора, вычисляется по формуле $T^t = E_{av}^t + 3\sigma_{av}^t$. Если оказывается, что SAD больше порога, то включается процедура иерархического поиска. Иначе вектор считается найденным правильно, и поиск прекращается (Рис. 1).

2.6 Иерархический поиск

Метод иерархической компенсации состоит в том, что мы уменьшаем кадр в 4 раза по каждому измерению простым усреднением значений пикселей (box-фильтр) и делаем полный перебор с шагом в 2 пикселя по этим уменьшенным кадрам. Затем уточняем вектор с пиксельной точностью по уменьшенным кадрам. Таким образом, мы получаем вектор с точностью до 4 пикселей, который уточняется при помощи вышеописанной процедуры допоиска.

2.7 Заполнение НВК следующего кадра

Для каждого найденного вектора в текущем кадре мы находим его продолжение в следующий кадр. После этого определяем, в какой блок он попадает, и добавляем этот вектор в качестве кандидата для этого блока и его 8 соседей. Этот алгоритм основан на предположении, что объекты в видеопоследовательности движутся равномерно. Поэтому, если найти положение объекта в следующем кадре, то в этом месте вектор движения будет такой же, как и в предыдущем кадре для данного объекта.

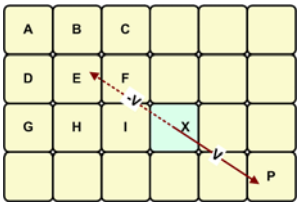

Рис. 4. Заполнение НВК следующего кадра

Рассмотрим пример на Рис. 4. Пусть X – текущий блок. Для него был найден вектор V. Значит, объект из предыдущего кадра двигался в текущий кадр с вектором –V. Если объект сохранит вектор своего движения, то в следующем кадре он будет находиться в блоке E. Но поскольку движение объекта может меняться, то вектор на всякий случай также добавляется в качестве кандидата в блоки с A по I.

3. СРАВНЕНИЕ

Предложенный алгоритм был реализован в виде плагина к программе VirtualDub (www.virtualdub.org). Также в данном плагине был реализован ряд широко известных алгоритмов, с которыми было проведено сравнение. Кроме них для сравнения была взята утилита MVTools (www.avisynth.org.ru) для программы AVISynth (www.avisynth.org).

Предложенный алгоритм запускался со следующими настройками: блоки 8×8, tD = 4, k_E = k_σ = 0.3, блоки иерархической компенсации 8x8 (32x32 в масштабе исходного кадра), область поиска ±32 пикселя.

Из широко известных алгоритмов был реализован шаблонный поиск с уменьшением размера шаблона на каждом шаге [9], несколько вариантов E3DRS [2], и иерархический поиск [3]. Варианты E3DRS отличаются только алгоритмом уточнения вектора: при помощи 4-х или 8-точечного шаблона. Буквы “pel” после названия алгоритма обозначают, что уточнение было только с пиксельной точностью.

Сравнение алгоритмов производилось на 6 видеопоследовательностях:

Номер	Название	Разрешение	Кол-во кадров
0	bus.avi	352×288	150
1	foreman.avi	352×288	300
2	mobl.avi	704×576	745
3	nba.avi	320×240	488
4	Schumacher.avi	720×576	219
5	tennis.avi	704×576	745

Результаты сравнения предложенного алгоритма с простейшими методами приведены на Рис. 5. PSNR приводится в виде разности с PSNR полного перебора с четвертьпиксельной точностью в области ±32 пикселя. Время работы приведено в виде отношения ко времени работы предложенного алгоритма. Выделенные точки соответствуют замеру на одной из последовательностей.

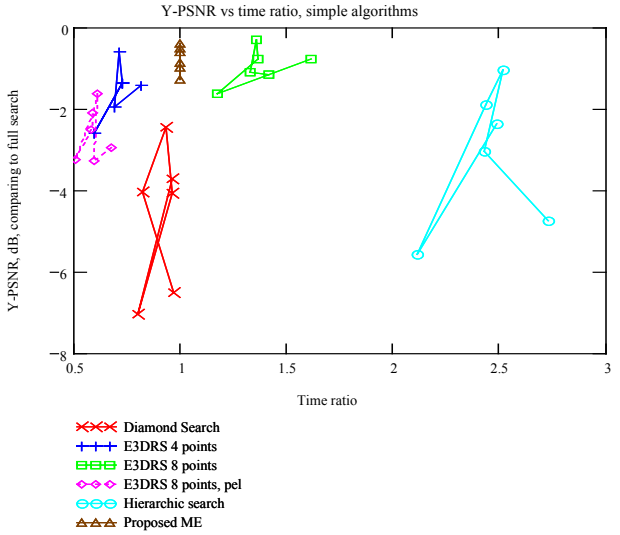

Рис. 5. Y-PSNR и время работы, сравнение с широко известными алгоритмами

Далее на Рис. 6 и Рис. 7 приведены результаты сравнения с утилитой MVTools. Графики полностью аналогичны графику сравнения с широко известными алгоритмами. Поскольку автор MVTools затруднился предложить наилучшие настройки, то было проведено 4 замера с различными пресетами, чтобы наиболее полно оценить возможности данной утилиты. Отличия заключались в вызове функции MVAnalyse, были использованы следующие параметры:

Пресет	Параметры
MVTools fast	pel = 2, isb = false
MVTools slow	pel = 2, isb = false, search = 3, searchparam = 32
MVTools fast, true motion	pel = 2, isb = false, truemotion = true
MVTools fast, slow motion	pel = 2, isb = false, truemotion = true, search = 3, searchparam = 32

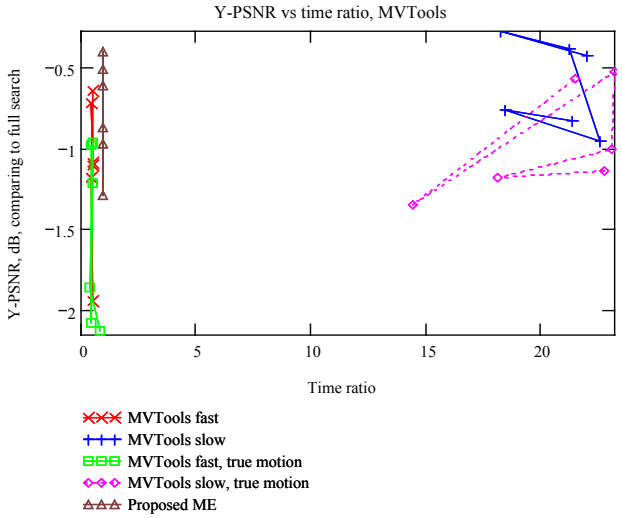

Рис. 6. Y-PSNR и время работы, сравнение с MVTools

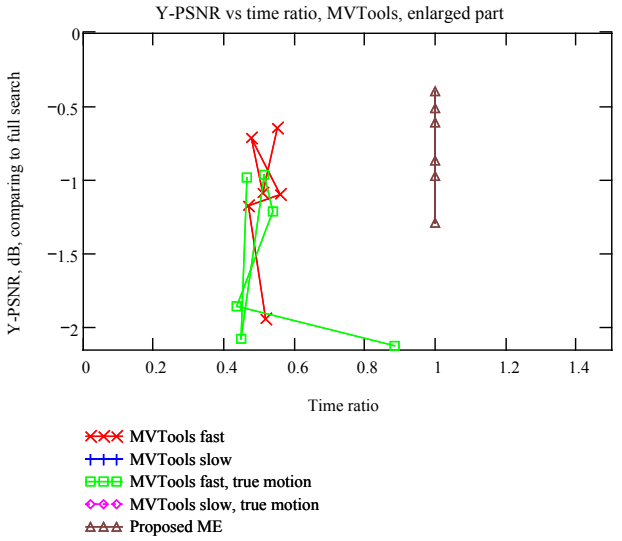

Рис. 7. Время работы, сравнение с MVTools, увеличена левая часть

4. ДАЛЬНЕЙШИЕ ПЛАНЫ

В дальнейшие планы входит улучшение качества и скорости алгоритма. Также будет проведено сравнение алгоритма с более современными алгоритмами нахождения движения, такими как MVFAST [3], PMVFAST [4], FAME [5], компенсация из кодека x264 (http://developers.videolan.org/x264.html). Поскольку алгоритм планируется использовать в кодировании видео, то

необходимо будет привести его в соответствие со стандартом H.264 [8]. Кроме этого, алгоритм также планируется использовать в обработке видео, а значит, необходимо находить настоящие вектора движения, а не минимизирующие SAD. Тем самым, надо будет разработать две ветви алгоритма. Для каждой ветви должна быть отдельная методика оценки качества.

5. ЗАКЛЮЧЕНИЕ

В данной статье был представлен новый алгоритм компенсации движения в видеопоследовательностях. В данном алгоритме были реализованы такие идеи, как решение о сложности движения по характеру НВК, фильтрация НВК, нахождение областей с новым движением, использование адаптивного порога и использование векторов текущего кадра для формирования НВК следующего кадра. Также было проведено сравнение с другими алгоритмами, которое показало, что текущая реализация всех вышеперечисленных идей перспективна, но все еще требует доработок.

В заключение хочется выразить благодарность своему научному руководителю Ватолину Д.С. за идеи, помощь в написании статьи и моральную поддержку, аспиранту нашей лаборатории Лукину А.С. за советы и моральную поддержку, заведующему нашей лабораторией Баяковскому Ю.М. за помощь в поиске материалов по компенсации движения в видео последовательностях, а также Александру Паршину и Станиславу Солдатову за усидчивость при проверке данного документа.

6. БИБЛИОГРАФИЯ

[1] International Organization for Standardization, *Short MPEG-2 description, http://www.chiariglione.org/MPEG/standards/mpeg-2/mpeg-2.htm*

[2] S. Olivieri, L. Albani, G. de Haan, *A low-complexity motion estimation algorithm for H.263 video coding, Proc. Philips Conf. on Digital Signal Processing, Nov. 1999, paper 17.3, Veldhoven, http://www.ics.ele.tue.nl/~dehaan/pdf/49_DSP99_E3DRS.pdf.*

[3] Chia-Wen Lin, Yao-Jen Chang, Yung-Chang Chen, *Hierarchical Motion Estimation Algorithm Based on Pyramidal Successive Elimination, International Computer Symposium 1998, October 1998*, *http://www.cs.ccu.edu.tw/~cwlin/pub/ics98.pdf.*

[4] P. I. Hosur and K. K. Ma, *Motion vector field adaptive fast motion estimation, Second Int. Conf. Inf., Commun., Signal Process., Singapore, Dec. 1999.*

[5] A. M. Tourapis, O. C. Au, and M. L. Liou, *Fast block-matching motion estimation using predictive motion vector field adaptive search technique (PMVFAST), ISO/IEC JTC1/SC29/WG11 MPEG2000/M5866, Noordwijkerhout, The Netherlands, Mar. 2000.*

[6] Ishfaq Ahmad, Weiguo Zheng, Jiancong Luo, Ming Liou, A *Fast Adaptive Motion Estimation Algorithm, IEEE Transactions on circuits and systems for video technology, vol. 16, No. 3, March 2006*

[7] Shih-Yu Huang, Chuan-Yu Cho, and Jia-Shung Wang, *Adaptive Fast Block-Matching Algorithm by Switching Search Patterns for Sequences With Wide-Range Motion Content, IEEE on circuits and systems for video technology, vol. 15, No. 11, November 2005*

[8] Thomas Wiegand, Gary Sullivan, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG 7th meeting, Pattaya, Thailand, 7-14 March, 2003*

[9] Nam Ling, Rajesh Advani, *A High Speed Motion Estimator Using 2-D Log Search Algorithm, http://csdl2.computer.org/comp/proceedings/dcc/1996/7358/00/73580448.pdf*

Об авторах

Сергей Юрьевич Путилин – аспирант факультета вычислительной математики и кибернетики Московского Государственного Университета им. М.В. Ломоносова

Адрес: Москва, 119992, Воробьевы горы, МГУ, 2-й учебный корпус, факультет ВМиК, кафедра Автоматизации Систем Вычислительных Комплексов, лаборатория Графики и Мультимедиа.

E-mail: sputilin@graphics.cs.msu.ru

Fast motion estimation algorithm for video processing

Abstract

The paper is devoted to block-based motion estimation algorithm for video processing. Algorithm's main features are adding of candidate motion vectors for the next frame, detection of areas with new objects and hierarchical search in these areas. Proposed algorithm is compared by speed and quality with wide-known simple algorithms and MVTools plugin for AVISynth.

Keywords*: motion estimation, video processing, video coding.*

About the author

Sergey Putilin is a Ph.D. student at the Laboratory of Graphics and Multimedia at the Moscow State University, Department of Computational Mathematics and Cybernetics.

E-mail: sputilin@graphics.cs.msu.ru