УДК 004.627:004.932.2

Алгоритмы сжатия экранного видео, использующие корреляцию соседних кадров

Д.В. Дружинин

ООО «Квантум-Софт» (Томск, Россия)

Screen Video Compression Algorithms Based on Neighbor Frames Correlation.

D.V. Druzhinin

LLC "Quantum Soft" (Tomsk, Russia)

Рассмотрены ключевые отличия экранного видео от камерного. Делается вывод о необходимости адаптации алгоритмов сжатия видео для обработки экранного видео. Представлены два алгоритма, основанных на сравнении изображений и адаптированных для сжатия экранного видео. Первый из этих алгоритмов находит блоки, в которых есть изменившиеся относительно предыдущего кадра пиксели. Второй алгоритм определяет номера строк и столбцов, на пересечении которых находятся изменившиеся пиксели. Проводится аналитическое и практическое сравнение этих алгоритмов при различных изменениях на экране, произошедших в течение временного интервала, разделяющего два кадра. Представлена классификация типов движений в экранном видео, модификация алгоритма оценки движения, разработанного для обработки экранного видео, выявляющая наиболее часто встречающиеся типы движений и отличающаяся от аналогичных по назначению разработок значительно большей скоростью выполнения. Приведены результаты тестирования реализаций различных модификаций алгоритма оценки движения. Представлена техника предварительного сравнения диагональных элементов блоков, которая позволяет достичь дополнительного ускорения при выполнении оценки движения при сжатии экранного видео.

Ключевые слова: экранное видео, алгоритмы сжатия, сжатие без потерь, оценка движения.

Key distinctions between screen and camera videos are investigated in this paper, and a conclusion to adapt video compression algorithms for screen video processing is drawn in accordance to the distinctions. Two algorithms based on image comparison and adapted for screen video processing are proposed. The first algorithm searches for blocks with pixels changed from previous video frame. The second algorithm locates row and column intersections with changed pixels. Practical and analytical comparison of the two algorithms under various pixel changes within two video frames are provided. A classification of movement types in a screen video is proposed, and movement detection algorithm modification for screen video compression is developed. The modified algorithm allows detection of the most common movement types and has significantly greater processing speed than similar algorithms. Diagonal block elements preliminary comparison technique for faster movement detection is developed for screen video processing. Test results of various movement detection algorithm modifications are presented.

Key words: screen video, compression algorithms, lossless compression, movement detection.

DOI 10.14258/izvasu(2014)1.2-14

Видеокадры можно разделить на ключевые, сжимаемые независимо от других кадров, и промежуточные, сжатие которых производится с использованием информации о других кадрах.

Экранное видео формируется путем периодического снятия скриншотов экрана. Часто сжатие и запись на жесткий диск этого типа видеоданных требуется осуществлять в режиме реального времени. С учетом того, что экранное видео — высокого разрешения, актуальна задача разработки быстрых алгоритмов сжатия экранного видео. Экранное видео в значительной степени отличается от снятого видеокамерой. Камерное видео запечатлевает объекты реального мира с определенной точки обзора при определенном

освещении. Для этого типа видеоданных характерны плавные (непрерывно-тоновые) цветовые переходы между соседними пикселями. Экранное видео запечатлевает искусственным образом созданные объекты, для которых характерны резкие (дискретно-тоновые) цветовые переходы между соседними пикселями. Разумно использовать информацию об особенностях экранного видео, чтобы адаптировать алгоритмы сжатия, использующие корреляцию соседних кадров, для обработки экранного видео.

1. Алгоритмы сжатия экранного видео, основанные на сравнении изображений. Описанные ниже два алгоритма сжатия видео, основанные на сравнении изображений, были разработаны автором на основе общих идей сжатия видео, изложенных в [1].

1.1. Алгоритм, основанный на попиксельном сравнении изображений. Для каждого промежуточного кадра в выходной файл записываются номера строк и столбцов, в которых есть изменившиеся пиксели относительно ключевого кадра, а затем цвета пикселей, находящихся на пересечении этих строк и столбцов. Остальную часть промежуточных кадров можно восстановить по ключевому кадру. Это алгоритм сжатия без потерь информации, обладающий линейной трудоемкостью. Входные данные этой операции (назовем ее попиксельный xor) — два изображения одинакового размера. Выходные данные — набор элементов, количество которых равно количеству пикселей во входном изображении. Каждый такой элемент является индикатором равенства или неравенства цветов двух пикселей.

1.2. Алгоритм, основанный на поблоковом сравнении изображений. Отличие этого алгоритма от того, который основан на попиксельном сравнении, заключается в том, что в выходной файл записываются те блоки пикселей промежуточного кадра, в которых есть хотя бы один изменившийся пиксель относительно соответствующего блока ключевого кадра. Также в выходной файл записываются номера изменившихся блоков.

Рассмотрим принципы выбора размера блока. При увеличении размера блока уменьшаются накладные расходы, связанные с хранением номеров изменившихся блоков, но уменьшается и точность определения изменившейся области (и наоборот). Был выбран размер блока 8 * 8 пикселей, так как при таком размере блока на большинстве тестов была достигнута максимальная степень сжатия. Входные данные те же, что и при попиксельном xor. Существенное отличие состоит в значительно меньшем (в $2^6 = 8 * 8$ раз) размере выходного массива, так как каждый его элемент является индикатором равенства или неравенства двух блоков пикселей.

1.3. Сравнение алгоритмов, основанных на попиксельном и поблоковом xor между собой. Оба описанных выше алгоритма выполняются за близкое время. Было установлено, что алгоритм, основанный на попиксельном сравнении, демонстрирует более высокую степень сжатия при таком изменении промежуточного кадра относительно ключевого, где пользователь сворачивает окно или, наоборот, открывает свернутое (табл. 1). В этом случае алгоритм, основанный на попиксельном сравнении, выявит в точности ту область, которая была изменена. А алгоритм, основанный на поблоковом сравнении, будет считать измененной область, охватывающую реально изменившуюся область, так как блок 8 * 8 пикселей считается измененным, если изменился хотя бы один пиксель в блоке. Таким образом, по периметру реально изменившейся области будет захвачена «буферная зона» толщиной максимум в семь пикселей из неизменившейся области.

Пусть $T_{ср} = T_{max} / 2 = 7 / 2 \approx$ (примерно равно) 4, где $T_{ср}$ — это толщина буферной зоны в среднем, а T_{max} — максимальная толщина буферной зоны. Тогда среднее количество неизменившихся пикселей $N_{ср}$ из «буферной зоны» вокруг изменившейся области, которые алгоритм, основанный на поблоковом сравнении, будет считать изменившимися, можно посчитать по следующей формуле:

$$N_{ср} = P * T_{ср} + const,$$

где P — это периметр реально изменившейся области, измеряемый в пикселях; const — некоторая небольшая константа, определяемая формой изменившейся области.

Алгоритм, основанный на поблоковом сравнении, показывает более высокий коэффициент сжатия при таком изменении промежуточного кадра относительно ключевого, где пользователь перемещает окно на некоторое расстояние по направлению, близкому к диагональному (табл. 1). Для определенности предположим, что окно было перемещено сверху вниз и слева направо (рис.). В этом случае алгоритм, основанный на попиксельном сравнении, будет считать изменившейся областью прямоугольник, координаты верхнего левого угла которого совпадают с координатами левого верхнего угла окна на ключевом кадре (до движения), а координаты правого нижнего угла совпадают с координатами правого нижнего угла окна на промежуточном кадре (после движения). Назовем такую область охватывающим прямоугольником. На рисунке это прямоугольник с вершинами (x_{11}, y_{11}), (x_{22}, y_{21}), (x_{23}, y_{23}), (x_{14}, y_{24}). Количество пикселей N в неизменившейся области, которая попадает в охватывающий прямоугольник, можно посчитать по следующей формуле:

$$N = (x_{22} - x_{12}) * (y_{12} - y_{22}) * 2.$$

Размер «буферной зоны», захватываемой алгоритмом, основанным на поблоковом сравнении,

в большинстве случаев значительно меньше той неизменившейся области, которая попадает в охватывающий прямоугольник.

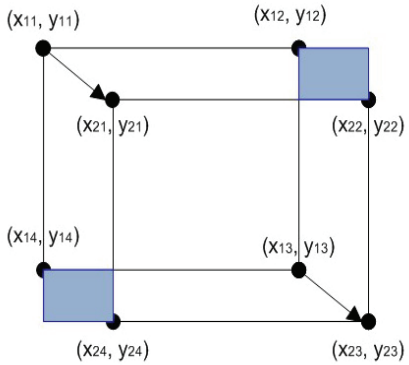

Рис. Перемещение окна по направлению, близкому к диагональному окну до движения соответствует прямоугольник с вершинами (x_{11}, y_{11}), (x_{12}, y_{12}) (x_{13}, y_{13}) (x_{14}, y_{14}). Окну после движения соответствует прямоугольник с вершинами (x_{21}, y_{21}), (x_{22}, y_{22}) (x_{23}, y_{23}) (x_{24}, y_{24}). Серым цветом обозначены неизменившиеся области, которые попадают в охватывающий прямоугольник

При тестировании каждый кадр имел разрешение 1024*768 и глубину цвета в 32 бита. Это сравнение показывает, что ни один из рассмотренных выше алгоритмов не является безусловно лучшим по степени сжатия.

Таблица 1

Результаты тестирования алгоритмов, основанных на попиксельном и поблоковом сравнении при сворачивании окна и перемещении окна по диагонали

Тип изменения экрана / алгоритм	**Попиксельное сравнение, Б**	**Поблоковое сравнение, Б**
Сворачивание окна	921600	997632
Перемещение окна по диагонали	673554	476928

2. Алгоритм оценки движения, адаптированный для обработки экранного видео. В [2] представлен алгоритм оценки движения, адаптированный для обработки экранного видео. Рассмотрим более подробно отличия экранного видео от камерного, которые наиболее ярко проявляются при использовании алгоритма оценки движения, а также особенности алгоритма [2], использующие эти свойства.

1. Для экранного видео характерны низкая степень совпадения для большинства векторов движения и полное (или почти полное) совпадение для единственного вектора, в то время как для камерного видео часто существуют несколько векторов движения, которые обеспечивают высокую степень сходства соответствующих блоков по заданному критерию. Поэтому в [2] используется расстояние Хэмминга для измерения степени сходства блоков, т.е. рассчитывается количество пар соответствующих пикселей одного цвета.

2. В экранном видео объекты могут перемещаться на значительное расстояние за временной отрезок, разделяющий два соседних кадра, в то время как в видео, снятом видеокамерой, обычно происходит плавное движение объектов от кадра к кадру. Поэтому в [2] используется более обширная область поиска (вплоть до поиска по всему кадру), чем при сжатии камерного видео.

3. Градиентный метод поиска оптимального вектора движения активно применяется при сжатии камерного видео [3, 4]. Но в случае экранного видео в силу дискретно-тоновой природы таких видеоданных выбор из нескольких ближайших векторов движения в приоритетном порядке в качестве текущего вектора v1, который обеспечивает минимальное расхождение (если оно не равно 0), зачастую неэффективен. Поэтому в [2] не используется градиентный метод.

4. В экранном видео обычно значительная часть соответствующих пикселей соседних кадров совпадают. Поэтому в [2] предлагается сначала провести попиксельное сравнение текущего и предыдущего кадров на равенство. Затем оценка движения проводится только для блоков текущего кадра, изменившихся относительно предыдущего кадра.

5. В экранном видео велика вероятность соседства нескольких блоков, имеющих один и тот же вектор движения (например, при движении окна), в отличие от камерного видео, где одинаковый вектор движения для соседних блоков встречается несколько реже за счет изменения формы объекта и угла обзора при движении этого объекта.

2.1. Описание алгоритма оценки движения, адаптированного для экранного видео. Выявление всех движений в экранном видео оказалось слишком трудоемкой задачей для выполнения в режиме реального времени.

В такой ситуации логично провести некоторую классификацию типов движений в экранном видео и разработать алгоритмы, выявляющие некоторые из этих типов движений и работающие при этом значительно быстрее алгоритма, выявляющего все типы движений.

Можно выделить три основных вида движений в экранном видео:

1) движения по вертикали и горизонтали (потенциально на большие расстояния), осуществляемые вследствие вертикального и горизонтального скроллинга, нажатия пользователем на клавиши вниз, вверх, вправо, влево, Pg Up, Pg Dn и пр.

2) движения в произвольном направлении на небольшие расстояния, осуществляемые вследствие, например, достаточно плавного перетаскивания пользователем окна;

3) движения, осуществляемые в произвольном направлении на большие расстояния.

Как правило, подавляющее большинство движений различных объектов на экране, возникающих в ходе работы пользователя, относятся к (1) либо (2) категории. Использовав приведенную классификацию движений в экранном видео, удалось разработать следующую схему кодирования на основе алгоритма оценки движения, описанного в [2]. Последовательно выполняются две модификации алгоритма оценки движения:

1) алгоритм, осуществляющий поиск блока, соответствующего текущему блоку, только по вертикали и по горизонтали;

2) алгоритм, осуществляющий поиск блока, соответствующего текущему блоку, во всех направлениях, но только в ближайшей окрестности текущего блока.

Для ускорения работы алгоритма оценки движения при обработке экранного видео предлагается использовать технику предварительного сравнения диагональных элементов. Только в случае когда все диагональные элементы попарно равны между собой, происходит сравнение всех элементов этих блоков. Такая техника приводит к ускорению выполнения алгоритма оценки движения, так как она одновременно учитывает и горизонтальную, и вертикальную корреляцию пикселей движущегося объекта.

Вначале производится вычисление минимального прямоугольника, охватывающего все изменившиеся относительно предыдущего кадра области текущего кадра, с помощью одного из представленных выше алгоритмов, основанных на сравнении кадров. Именно в пределах этого прямоугольника в дальнейшем будет осуществляться поиск соответствия для текущего блока.

2.2. Результаты тестирования. При тестировании был выбран размер блока 16 * 16 пикселей. Максимальные отклонения по оси x и y для алгоритмов-участников тестирования (5) и (6) рассчитывались по формуле n * blockSize, где blockSize — это размер блока. При этом было выбрано значение n, равное 3.

При тестировании каждый кадр имел разрешение 1024*768 и глубину цвета в 32 бита. Тестирование проводилось на платформе со следующими характеристиками: процессор Intel Core 2 Duo E6750 2,66 ГГц, оперативная память DDR2 2Гб, операционная система Windows XP.

Замечание 1: замеряемый параметр «Количество блоков» означает количество блоков, для которых было найдено соответствие данной реализацией. При этом суммарное количество блоков равно 1024 * 768 / (16 * 16) = 3072. Надо учитывать, что некоторые блоки не изменились по сравнению с предыдущим кадром.

Замечание 2: запись 3 → 5 означает, что текущий кадр сначала обрабатывается реализацией (3) алгоритма оценки движения, а затем реализацией (5).

Рассмотрим результаты тестирования при двух типах движений (табл. 2, 3).

Таблица 2

Движения первого типа по классификации, приведенной выше (для возникновения движения такого типа использовался скроллинг)

Алгоритм / Параметр	**Время выполнения, мс**	**Количество блоков, шт.**
1. Алгоритм оценки движения, представленный в [2]	10049	1436
2. Алгоритм оценки движения, представленный в [2], с добавленной начальной проверкой диагональных элементов	4012	1436
3. Модификация (1) алгоритма оценки движения, представленная в данной работе (поиск блока, соответствующего текущему блоку, осуществляется только по вертикали и по горизонтали)	24	1436
4. Модификация (1) алгоритма оценки движения, представленная в данной работе, без начальной проверки диагональных элементов	65	1436
5. Модификация (2) алгоритма оценки движения, представленная в данной работе (поиск блока, соответствующего текущему блоку, осуществляется во всех направлениях, но только в ближайшей окрестности текущего блока)	510	697
6. Модификация (2) алгоритма оценки движения, представленная в данной работе, без начальной проверки диагональных элементов	1094	697
3 → 5	24 → 10	1436 → 0
5 → 3	510 → 15	396 → 830

Таблица 3

Движения второго типа по классификации, приведенной выше (для возникновения движения этого типа использовалось плавное перетаскивание окна)

Алгоритм / Параметр	**Время выполнения, мс**	**Количество блоков, шт.**
1. Алгоритм оценки движения, представленный в [2]	3561	1384
2. Алгоритм оценки движения, представленный в [2], с добавленной начальной проверкой диагональных элементов	2422	1384
3. Модификация (1) алгоритма оценки движения, представленная в данной работе (поиск по вертикали и по горизонтали)	63	450
4. Модификация (1) алгоритма оценки движения, представленная в данной работе, без начальной проверки диагональных элементов	140	450
5. Модификация (2) алгоритма оценки движения, представленная в данной работе (поиск в ближайшей окрестности текущего блока)	47	1384
6. Модификация (2) алгоритма оценки движения, представленная в данной работе, без начальной проверки диагональных элементов	62	1384
3 → 5	62 → 47	463 → 962
5 → 3	47 → 16	1379 → 38

Быстрее всего выполняется реализация (3), что позволяет использовать ее при сжатии экранного видео в режиме реального времени. Но реализация (3) не в состоянии выявить движения второго типа, поэтому имеет смысл рассмотреть комбинации реализаций (3) и (5). Наиболее стабильное время выполнения демонстрирует последовательность (3) → (5). Время работы этой последовательности не превышает 109 мс. При этом последовательно выполняемые реализации (3) и (5) находят соответствие для практически такого же количества блоков, что и реализация (1).

Заключение. Предварительное вычисление минимального прямоугольника, охватывающего все изменившиеся относительно предыдущего кадра области текущего кадра, с помощью одного из представленных алгоритмов, основанных на сравнении кадров, позволяет ускорить оценку движения. Проведение классификации движений в экранном видео сделало возможным поиск заданных типов движений. Предложенная схема сжатия позволяет ускорить выполнение оценки движения десятикратно при незначительных потерях в количестве распознанных движений, так как большая часть движений в экранном видео относятся именно к (1) или (2) типам. Поэтому такой алгоритм оценки движения может быть использован на практике.

Библиографический список

1. Сэломон Д. Сжатие данных, изображений и звука / пер. с англ. В.В. Чепыжова. — М., 2006.

2. Motion Estimation / Compensation for Screen Capture Video [Электронный ресурс] // FreePatentsOnline.com. — URL: http://www.freepatentsonline.com/7224731.html

3. Movement Estimation System for Video Signals Using a Recursive Gradient Method [Электронный ресурс] / FreePatentsOnline.com. — URL: http://www.freepatentsonline.com/4695882.html

4. Keller Y., Averbuch A. Fast Motion Estimation Using Bidirectional gradient Methods. [Электронный ресурс]. — URL: http://www.eng.biu.ac.il/_kellery1/publications/pdf/optical_flow_ieee_final.pdf