А. В. Ткачев¹, И. Г. Таранцев²

¹ *Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия*

² *Институт автоматики и электрометрии СО РАН
пр. Академика Коптюга, 1А, Новосибирск, 630090, Россия*

alexander@tkachov.ru, i.tarantsev@g.nsu.ru

АВТОМАТИЧЕСКОЕ ОПРЕДЕЛЕНИЕ ПОРЯДКА ПОЛЕЙ В ПОСЛЕДОВАТЕЛЬНОСТИ ИЗОБРАЖЕНИЙ С ЧЕРЕССТРОЧНОЙ РАЗВЕРТКОЙ

В том случае, когда видео с чересстрочной разверткой воспроизводится с неправильным порядком полей, зрители ощущают дискомфорт, если в видео присутствуют движения относительно больших контрастных объектов. В статье описывается алгоритм автоматического определения порядка полей в последовательности изображений с чересстрочной разверткой, позволяющий точно определять порядок полей, с которым необходимо воспроизводить конкретную последовательность, чтобы зрители не ощущали дискомфорта.

Ключевые слова: порядок полей, чересстрочная развертка, прогрессивная развертка.

Введение

В телевидении используется чересстрочная развертка, когда кадр собирается из двух полей, снимаемых в разное время. В одно поле входят все четные строки кадра (нижнее поле), в другое – все нечетные (верхнее поле). Если динамичное видео, снятое с одним порядком полей, просматривать с другим порядком полей, то будет замечен дефект, сильно напрягающий человеческий глаз. Алгоритмы автоматического определения исходного порядка полей в программах для редактирования видео ориентируются не на изображение, а на служебные поля в видеофайле, поэтому программа может неправильно определить исходный порядок полей. Поскольку современные программы редактирования видео достаточно редко ошибаются в определении исходного порядка полей, то операторы начинают пренебрегать обязательным просмотром результирующего видеофайла на телевизоре с чересстрочной разверткой. В результате в эфире зрители видят сюжеты с перепутанным порядком полей, что регулярно случается даже на федеральных каналах, например на канале «Вести 24». Чтобы исключить подобные ошибки, необходимо использовать автоматизированное средство контроля корректности определения порядка полей, позволяющее без участия оператора определить исходный порядок полей и, если он не совпадает с указанным в файле, предупредить об этом оператора.

Цель работы – создать алгоритм, позволяющий в автоматическом режиме определять исходный порядок полей. При этом время работы алгоритма должно быть достаточно малым – сравнимым со временем просмотра ролика оператором на телевизоре с чересстрочной разверткой. Иначе говоря, время обработки конкретного ролика должно быть сравнимо с длительностью этого ролика.

Ткачев А. В., Таранцев И. Г. Автоматическое определение порядка полей в последовательности изображений с чересстрочной разверткой // Вестн. НГУ. Серия: Информационные технологии. 2017. Т. 15, № 2. С. 74–84.

Необходимо отметить, что проблема дрожания проявляется тем сильнее, чем крупнее объект на изображении, чем больше скорость его перемещения и чем контрастнее этот объект на фоне других. Если объект небольшой, медленно движется или его можно визуальнo принять за часть фона, то дрожание в последовательности с неправильным порядком полей заметно не будет. Из этих соображений вытекает следующее: необходимо обеспечить нахождение последовательностей, в которых проблема дрожания проявляется очень сильно, а последовательностями, в которых проблема человеческому глазу незаметна, можно пренебречь.

Анализ существующих решений

В открытом доступе был найден только один алгоритм, направленный на определение порядка полей. Кроме того, был обнаружен алгоритм, определяющий наличие или отсутствие «эффекта чересстрочности». Также существует простое и очевидное решение, заключающееся в анализе векторов смещения. Рассмотрим эти алгоритмы подробнее.

Алгоритм 1. Анализ мер движения при разных порядках полей¹. Мерой движения является сумма попиксельных отличий соседних полей:

$$\text{diff}(t) = \frac{1}{2} \sum_{i,j} |Y_{i,j,t+1} - Y_{i,j,t}| + \frac{1}{4} \sum_{i,j} |U_{i,j,t+1} - U_{i,j,t}| + \frac{1}{4} \sum_{i,j} |V_{i,j,t+1} - V_{i,j,t}|,$$

где

t – номер изображения,

i, j – координаты пикселя изображения,

Y, U, V – значения соответствующих каналов изображения.

Оценив величину движения при разных порядках полей, данный алгоритм выбирает тот порядок, при котором эта величина меньше. Поскольку diff соответствует величине движения, то последовательность с меньшими значениями будет соответствовать тому порядку полей, при котором движение происходит более плавно.

Алгоритм 2. Сравнение корреляций соседних строк в кадре [1]. Данный алгоритм сравнивает корреляцию соседних строк (четных с нечетными) и строк, взятых через одну (четных с четными и нечетных с нечетными). Сравнивается небольшое число точек (1–10 %), которые выбираются псевдослучайным образом из всего изображения.

$$\begin{cases} I_1 = \sum_{k=0}^N (Y_{i(k),j(k)} - Y_{i(k),j(k)+1})^2 \\ I_2 = \sum_{k=0}^N (Y_{i(k),j(k)} - Y_{i(k),j(k)+2})^2 \end{cases}.$$

Здесь

k – номер точки,

N – число анализируемых точек,

i, j – псевдослучайно выбранные координаты пикселя по номеру точки,

Y – значение канала яркости изображения.

Алгоритм вычисляет величину «эффекта чересстрочности», равную отношению I_1 к I_2 , для обоих порядков полей. Для определения меры движения будем использовать отношение I_1 к I_2 , посчитанных для всего изображения.

Алгоритм 3. Анализ полей векторов смещения при одном и другом порядке полей. При исходном порядке полей движущиеся объекты в любых двух соседних изображениях последовательности будут давать векторы смещения, направленные в одну и ту же сторону и примерно равные по величине. При неправильном порядке полей векторы смещения будут постоянно менять свое направление на противоположное и при этом будут сильно отличаться по величине (примерно в три раза). Такое поведение достаточно легко выделить программным способом, однако данное решение очень неэффективно, поскольку алгоритм поиска

¹ URL: <http://forum.doom9.org/showthread.php?t=104760>; http://avisynth.nl/index.php/Interlace_detection

Рис. 1. Ролик 1, чересстрочная развертка, сильное движение

Рис. 2. Ролик 2, чересстрочная развертка, слабое движение

Рис. 3. Ролик 3, прогрессивная развертка

векторов смещения сам по себе крайне трудоемок [2]. Такое решение подразумевает вычисление полей векторов смещения по два раза на каждую пару полей в последовательности, что не может выполняться на обычном процессоре с приемлемым быстродействием. Подобная разница в производительности наблюдается в алгоритмах кодирования (вычисляющих векторы смещения) и декодирования – кодирование ролика выполняется на несколько порядков медленнее декодирования. Поэтому в данной работе решено отказаться от вычисления векторов смещения.

Сравнение алгоритмов проводилось на нескольких эталонных роликах, ярко демонстрирующих типичные особенности изображений:

На рис. 1–3 приведены примеры эталонных роликов: с чересстрочной и прогрессивной разверткой, сильным и слабым движением, реальной и совмещенной съемкой, а также с синтезированным изображением.

На рис. 4 и 5 приведены графики, показывающие меры движения, полученные алгоритмами 1 и 2 для указанных роликов.

Видно, что разный порядок полей (TFF – верхнее поле первое, BFF – нижнее поле первое) дает разные по величине меры движения. Также нужно отметить, что для разных алгоритмов используется разная шкала меры движения. Для того чтобы сравнить алгоритмы между собой, будем использовать отношение большей меры движения к меньшей. Оно показывает, во сколько раз движение при одном порядке полей отличается от движения при другом. Чем выше данное отношение, тем с большей уверенностью можно говорить о том, какой порядок полей дает более плавное движение.

Рис. 4. Меры движения для роликов 1–3, полученные алгоритмом 1

Рис. 5. Меры движения для роликов 1–3, полученные алгоритмом 2

Рис. 6. Отношение мер движения для роликов 1–3

Как видно из графика на рис. 6, вычисленные алгоритмом 2 величины для разных порядков полей слабо отличаются (их отношение близко к единице), поэтому достоверно определить, какой порядок действительно дает меньшее дрожание, трудно. Алгоритм 1 показывает существенно более высокую чувствительность, потому алгоритм 2 был исключен из дальнейшего рассмотрения.

Проблемы существующего алгоритма и их решение

В процессе анализа и тестирования существующего алгоритма (алгоритм 1) был выявлен ряд особенностей обрабатываемых последовательностей. Эти особенности часто приводили к определению неправильного порядка алгоритмом.

«Жесткая склейка» – место резкой смены снимаемой сцены. В этом случае на одном кадре еще присутствует предыдущая сцена, а на следующем – уже совершенно другая (как, например, в ролике 1, показанном на рис. 1). Разница от движения даже крупных объектов гораздо меньше, чем разница, вычисленная между кадрами «жесткой склейки», поскольку изображение меняется целиком. Алгоритмы, которые не предполагают наличие «жестких склеек» в последовательностях изображений, получают в таких местах очень большие меры движения для обоих порядков полей (например, это видно на графиках для ролика 1 на рис 4 и 5). Из того что какая-то из мер движения немного меньше другой, не следует, что именно эта мера соответствует правильному порядку полей. Это значит, что надежнее проигнорировать места «жесткой склейки», чем пытаться определить по ним порядок полей.

Чтобы отбросить единичные выбросы (в частности, в местах «жесткой склейки») и таким образом уменьшить число ложных срабатываний, в конце обработки всей последовательности полученные значения отношений мер движения в кадре можно обработать медианным фильтром по семи кадрам. Этот фильтр ограничивает значение меры движения текущего кадра мерами движения первых двух и последних двух кадров в группе из семи кадров.

Наличие *контрастных мелких деталей* в соседних строках кадра приводит к тому, что даже в отсутствие движения в последовательности изображений между полями существует большая разница. Эта разница во многом случайна и может сильно изменяться от кадра к кадру из-за особенностей кодирования видео современными алгоритмами типа AVC и HEVC, когда первый кадр последовательности показывается достаточно грубо (сильно размытым), а затем за 2–3 кадра четкость повышается. Это мешает корректному сравнению, поскольку на фоне таких отличий небольшое изменение от движения объектов в кадре просто теряется. Чтобы снизить влияние статичных частей изображения, нужно обрабатывать не всё изображение, а только его движущиеся части.

Чтобы определить, какие пиксели являются движущимися, применяется алгоритм деинтерлейсинга, удваивающий частоту кадров. В результате его работы для каждого поля восстанавливается поле другой четности, соответствующее тому же моменту времени. Восста-

новленное поле совпадает по четности со следующим по времени полем исходной последовательности изображений, т. е. если было восстановлено верхнее поле, то и следующее поле будет верхним. Статические части изображений не изменяются от кадра к кадру, следовательно, разница между восстановленным и следующим полем должна быть минимальной. Если разница между пикселями превышает некоторый порог, то пиксель следует считать движущимся. В итоге получаем первичную маску движущихся пикселей.

Эффект дрожания хорошо заметен при движении крупных объектов, имеющих длинную непрерывную границу. Очевидно, что на границе движения таких объектов описанный алгоритм выделит изменившиеся пиксели. Движение очень маленьких объектов, площадь которых не превышает нескольких пикселей, не заметно зрителю. По разным причинам описанный алгоритм может выделить отдельные пиксели, не относящиеся к движению реальных объектов. Чтобы удалить подобные единичные группы пикселей, первичная маска обрабатывается фильтром типа «Эрозия», который оставляет центральный пиксель белым, если он изначально был белым и среди восьми его соседей есть как минимум два белых соседа. Получившееся бинаризованное изображение является маской для последующего сравнения изображений.

Кроме того, если движущихся пикселей немного, то, скорее всего, движения в последовательности почти нет, и кадр можно считать статичным. Если в маске число белых (движущихся) пикселей оказывается меньше, чем 0,5 % площади исходного изображения, то кадр считается статичным, и оценка движения в нем полагается равной нулю. В противном случае производится вычисление первичной меры движения, однако в сумму входят только те пиксели, которые в маске являются белыми. Получившееся значение является мерой движения в кадре для текущего порядка полей.

Порог по величине изменения, использующийся при вычислении первичной маски, и порог по минимальному числу белых пикселей в маске можно варьировать. При изменении одного из них для достижения тех же результатов нужно соответствующим образом изменить второй. Оба влияют на то, насколько слабые движения будут отбрасываться. Поскольку сильные движения не отбрасываются при любых разумных порогах, конкретные значения порогов не принципиальны.

Вертикальные движения камеры или объекта в последовательностях с прогрессивной разверткой (как, например, титры в конце фильма) приводят к ложным срабатываниям. Ошибка возникает из-за того, что при смещении изображения по вертикали на одну строку поля соседних кадров будут иметь более высокую корреляцию, чем поля внутри одного кадра, в результате чего алгоритм определит, что в ролике присутствует чересстрочная развертка, хотя изображение прогрессивное.

Чтобы устранить этот недостаток, в алгоритм был добавлен этап, на котором определяется, не является ли оцениваемое движение вертикальным движением в прогрессивных кадрах. Экспериментально было установлено, что алгоритм обнаруживает порядок TFF, если присутствует вертикальное движение вверх, а BFF – при вертикальном движении вниз. Соответственно, можно выполнять проверку смещения только в одном направлении. Алгоритм считает, что имеет место вертикальное движение при прогрессивной развертке, если мера движения, подсчитанная между кадрами при некотором смещении, существенно (не менее чем на 25 %) меньше меры движения, подсчитанной между полями одного кадра.

Иногда ролики содержат **незаметные проблемы**: небольшая тряска камеры, сильное, но очень короткое по времени движение или, наоборот, очень слабое движение на достаточно протяженном по времени участке. Такие проблемы зрители почти не замечают, а алгоритм может обнаружить в них неправильный порядок полей. Чтобы лишний раз не отвлекать оператора, нужно игнорировать такие небольшие проблемы.

Для этого в алгоритм добавлен этап подсчета интегральной меры движения в непрерывных последовательностях ненулевых значений. Если суммарная мера отличий в такой последовательности не превышает заданный порог, то вся последовательность игнорируется (обнуляется). Число игнорированных значений указывается в кратком сообщении для оператора, и они отмечаются на графике мер движения, чтобы оператор имел возможность проверить места, которые его заинтересуют.

Рис. 7. Вычисление мер движения

Общая схема работы алгоритма

В качестве первичной меры движения в разработанном алгоритме используется следующая величина:

$$\text{diff}(t) = \sum_{i,j} |Y_{i,j,t+1} - Y_{i,j,t}|.$$

Канал Y отвечает за яркость пикселя изображения. Дрожание хорошо заметно при любом цвете, поэтому для определения величины можно использовать только разность канала яркости. Символы i, j здесь указывают на координаты пикселей в поле, а t – на номер поля.

Алгоритм разбивается на четыре этапа.

1. Вычисление меры движения в кадре.
2. Проверка на вертикальное движение при прогрессивной развертке.

3. Обработка медианным фильтром.
4. Отсеивание коротких подпоследовательностей с небольшим движением.

Первые два этапа циклически выполняются для всех кадров последовательности, а затем выполняются третий и четвертый этапы. При этом второй этап проходят только те кадры, в которых было обнаружено движение.

Первый этап состоит из следующих частей.

1. Декодирование очередного кадра.
2. Разбиение текущего и двух предыдущих кадров на поля.
3. Подсчет меры движения для порядка полей TFF (верхнее поле первое).
4. Подсчет меры движения для порядка полей BFF (нижнее поле первое).
5. Вычисление отношения мер движения.
6. Выбор порядка полей по значению отношения.

На рис. 7 представлена схема первых четырех шагов первого этапа.

Проанализировав все полученные значения, алгоритм сообщает, какой порядок полей обнаружен в файле. Если в файле встречаются подпоследовательности с разным порядком полей, то алгоритм сообщает, что нужна ручная проверка файла оператором.

Тестирование алгоритма

На графике (рис. 8) приведено сравнение разработанного алгоритма с алгоритмами 1 и 2 на ролике 1. Видно, что сделанные модификации значительно увеличили чувствительность алгоритма.

Рис. 8. Сравнение вычисленного отношения мер движения разными алгоритмами

В качестве алгоритма деинтерлейсинга для тестирования разработанного алгоритма был выбран алгоритм деинтерлейсинга YADIF², обладающий высоким быстродействием и неплохим качеством восстановленного изображения. В статье Deinterlacing algorithms comparison³ приводится сравнение различных алгоритмов деинтерлейсинга. Для данной работы скорость обработки данных достаточно важна, поэтому был выбран один из самых быстрых алгоритмов. Из сравнения видно, что самыми быстрыми являются алгоритмы YADIF

² URL: <http://avisynth.org.ru/yadif/yadif.html>

³ URL: http://compression.ru/video/deinterlacer/deint_comparison.pdf

и TDeint. Остальные алгоритмы деинтерлейсинга во много раз медленнее. Алгоритм TDeint имеет очень много дополнительных параметров, в то время как у алгоритма YADIF они практически отсутствуют. Использование алгоритма YADIF позволяет выполнить полностью автоматическое тестирование, не требующее настроек деинтерлейсинга для каждого файла.

Чтобы оценить характеристики разработанного алгоритма, было проведено исследование на тестовой выборке, состоящей из 163 различных видеороликов, из которых 42 ролика с чересстрочной разверткой и сильным движением, 40 роликов с чересстрочной разверткой и слабым движением и 81 ролик с прогрессивной разверткой.

В табл. 1 представлены результаты обработки трех групп роликов.

Необходимо отметить, что практически все ошибки алгоритма приходятся на последовательности с чересстрочной разверткой и слабым движением. При просмотре таких роликов с неправильным порядком полей у зрителя не возникает дискомфорта. Алгоритм вычисляет отношение мер движения при разных порядках полей, но, поскольку движение слабое, данное отношение получается близко к единице, а такие ролики алгоритм отмечает как прогрессивные. Как видно из табл. 1, на прогрессивных роликах и роликах с чересстрочной разверткой и сильным движением алгоритм корректно определяет порядок полей в подавляющем большинстве случаев. Поэтому можно считать, что алгоритм успешно справляется со своей задачей: находит последовательности изображений, в которых проблема будет действительно заметна. Число конфликтных ситуаций, т. е. роликов, в которых алгоритм выявил несколько разных порядков полей, гораздо меньше, чем при обработке алгоритмом 1.

У алгоритма есть несколько параметров, влияющих на чувствительность к малым движениям. Изменяя их, можно добиться корректного определения порядка полей в роликах с малым движением, однако при этом из-за различного вида шумов алгоритм может обнаружить движение там, где его нет. Это может привести к увеличению числа ложных срабатываний и возникновению конфликтов. Каждый конфликт требует ручного просмотра, что увеличивает нагрузку на оператора.

Исследование также показало, насколько новый алгоритм замедлился из-за использования деинтерлейсинга и введения дополнительных этапов обработки. В среднем на обработку одного пикселя у алгоритма 1 уходило около 4 нс, а у нового – 26 нс.

При показе ролика с частотой 25 кадров в секунду на один кадр приходится 40 мс. Как видно из табл. 2, время обработки ролика новым алгоритмом начинает превышать время воспроизведения ролика при разрешении, большем SD. Однако тестируемая версия алгоритма работает только на одном ядре процессора, а сам алгоритм несложно модифицировать

Таблица 1

Результаты обработки последовательностей

Новый алгоритм, %	Алгоритм 1, %	Результат
С чересстрочной разверткой, сильное движение		
19	97,6	Верно определено
0	0	Ложные срабатывания
81	2,4	Обнаружено несколько типов движения
С чересстрочной разверткой, слабое движение		
42,5	67,5	Верно определено
22,5	27,5	Ложные срабатывания
35	5	Обнаружено несколько типов движения
С прогрессивной разверткой		
67,9	97,5	Верно определено
2,5	2,5	Ложные срабатывания
29,6	0	Обнаружено несколько типов движения

Таблица 2

Среднее время обработки одного кадра

Новый алгоритм, мс	Алгоритм 1, мс	
0,9	3,6	CIF ($< 640 \times 480$)
1,4	11	SD ($\sim 720 \times 576$)
7	57	HD ($\sim 1920 \times 1080$)
31	130	4K ($\sim 3840 \times 2160$)

для параллельной обработки, поскольку в нем практически нет зависимости по данным. В два рабочих потока очень просто вынести деинтерлейсинг с одним и с другим порядком полей, поскольку подавляющую часть времени занимает именно выполнение алгоритма деинтерлейсинга. Всё остальное (декодирование, выполнение этапов 2, 3 и 4) можно выполнить в основном потоке. Такая модификация даст существенный выигрыш и позволит обработать ролик с разрешением HD быстрее, чем при ручном просмотре. Для роликов с разрешением 4K проблема так остро не стоит, поскольку телевизионный стандарт для разрешений 4K пока не предусматривает наличие чересстрочной развертки (т. е. все ролики являются прогрессивными).

Заключение

В работе представлен алгоритм определения порядка полей, значительно превосходящий по своим характеристикам существующие. Несмотря на замедление из-за добавления новых этапов обработки, время работы его приемлемо и может быть снижено при помощи параллельной обработки данных. Надежность распознавания выше (97 % против 19 % на чересстрочных роликах с сильным движением), а практически все ложные срабатывания происходят только на роликах с небольшим движением, где проблема визуально незаметна. Иначе говоря, алгоритм позволяет решить основную проблему: в автоматическом режиме найти все ролики с чересстрочной разверткой, в которых проявляется дефект, сильно напрягающий человеческий глаз, если показывать ролик с неправильным порядком полей.

Разработанный алгоритм используется в приложении, автоматически обрабатывающем видеоролики в указанной директории. Оно существенно упрощает работу оператора, поскольку выделяет среди всех роликов только те, в которых могут быть проблемные места.

Список литературы

1. Слынько Ю. В. Комплекс алгоритмов улучшения изображений и компенсации их геометрических искажений. М., 2005. 100 с.
2. Солодушкин А. И., Кибиткин В. В., Плешанов В. С. Модифицированный алгоритм расчета поля векторов смещений для оценки деформации // Изв. Том. политехн. ун-та. 2011. Т. 318, № 5: Управление, вычислительная техника и информатика. С. 48–51.

A. V. Tkachev¹, I. G. Tarantsev²

¹ *Novosibirsk State University*

2 Pirogov Str., Novosibirsk, 630090, Russian Federation

² *Institute of Automation and Electrometry SB RAS*

1A Acad.Koptyug Str., Novosibirsk, 630090, Russian Federation

alexander@tkachov.ru, i.tarantsev@g.nsu.ru

AUTOMATIC FIELD ORDER DETECTION OF IMAGE SEQUENCES WITH INTERLACED SCANNING

When interlaced video is shown with wrong field order, motion of relatively big contrast objects looks shaky. Because of that viewers feel uncomfortable. This paper describes an algorithm designed to automatically detect field order. This algorithm allows to precisely detect correct field order for such video, so viewers won't see shaky motion.

Keywords: field order, interlaced video, progressive video.

References

1. Slyn'ko Iu. V. Kompleks algoritmov uluchsheniia izobrazhenii i kompensatsii ikh geometricheskikh iskazhenii [Set of algorithms improving images and compensating its geometric distortions]. Moscow, Moscow Institute of Physics and Technology, 2005. 100 p. (In Russ.)
2. Solodushkin A. I., Kibitkin V. V., Pleshanov V. S.. Modifitsirovannyi algoritm rascheta polia vektorov smeshchenii dlia otsenki deformatsii [Modified computation algorithm of displacement vector field for distortion estimation]. Tomsk, Izvestiia Tomskogo politekhnicheskogo universiteta T. 318, № 5 : Upravlenie, vychislitel'naia tekhnika i informatika [Tomsk Polytechnic University Bulletin V. 318, # 5: Control, computing hardware and computer science], 2011. P. 48–51. (In Russ.)