Математические структуры и моделирование 2017. № 3(43). С. 122–138

УДК 004.42
DOI: 10.25513/2222-8772.2017.3.122-138

ИССЛЕДОВАНИЕ СПОСОБОВ СОКРЫТИЯ ИНФОРМАЦИИ В ВИДЕОПОТОКЕ MPEG

Д.М. Бречка
к.т.н., доцент кафедры компьютерных технологий и сетей,
e-mail: dbrechkawork@yandex.ru
А.А. Литвиненко
студент, e-mail: a.litvinenko@outlook.com

Омский государственный университет им. Ф.М. Достоевского

Аннотация. Рассматриваются способы сокрытия информации в видеопотоке MPEG-4 с использованием кодека H.264. Для проверки возможности встраивания информации в видеопоток был выбран метод встраивания в коэффициенты дискретного косинусного преобразования. На основе кодека OpenH264 была реализована возможность встраивать и извлекать информацию.

Ключевые слова: кодек, MPEG-4, H.264, встраивание данных, видеопоток, сокрытие информации.

Введение

Технологии сокрытия информации — это методики, позволяющие поместить определённую информацию внутри другой цифровой информации (такой как текст, изображения, звук и т. д.). Основные области применения для технологий сокрытия информации — это обеспечение конфиденциальности передаваемой информации, защита авторских прав.

В зависимости от методов реализации и целей применения все технологии сокрытия информации делятся на технологии сокрытия данных и технологии цифровых водяных знаков. Сокрытие данных оперирует двумя наборами данных — носителем данных («обложкой») и встраиваемыми данными («сообщением»). Процесс сокрытия данных — это процесс встраивания «сообщения» в цифровую «обложку».

Целью данной работы является исследование процессов сокрытия и восстановления информации в видеопотоке MPEG-4 с использованием существующей реализации кодека H.264 с открытым исходным кодом.

1. Исследование стандарта MPEG-4

1.1. MPEG-4 и H.264

MPEG-4 — это формат медиаконтейнера, описанный в стандарте MPEG-4 Part 14 [1]. Медиаконтейнер MPEG-4 может использоваться как для хранения видео- и аудиопотока, так и для хранения других данных, например, субтитров и статичных изображений. Данные добавляются в контейнер в виде независимых потоков.

Самые распространённые форматы видео, поддерживаемые в MPEG-4, это:
- MPEG-4 Part 10 AVC/H.264;
- MPEG-4 Part 2 ASP;
- DivX/XviD.

Видеокодек H.264 в настоящее время является самым распространённым кодеком, используемым для кодирования видео в формате MPEG-4 [2]. Данный кодек позволяет гибко конфигурировать качество результирующего видео, предоставляя выбор между сжатием без потерь и сжатием с потерями.

1.2. Цветовое пространство

Видео в формате H.264 кодируется в цветовом пространстве YCbCr. Для представления каждого цвета YCbCr использует три значения: яркость (Y) и две цветоразностные компоненты — синюю (Cb) и красную (Cr). Само по себе цветовое пространство, по сути, является способом представления цветового пространства RGB и ничем принципиально от него не отличается. Но в отличие от RGB, YCbCr позволяет использовать особенность человеческого зрения в своих целях, произведя субдескритизацию цветоразностных компонент. Дело в том, что человеческий глаз воспринимает изменения в яркости сильнее, чем изменения в цветности. Использование YCbCr позволяет передавать информацию о яркости с полным, а цветоразностную информацию — с уменьшенным разрешением.

Структура дискретизации обозначается как соотношение $X : a : b$, где X — частота дискретизации яркостного сигнала (она же ширина макропикселя), a и b — число выборок цветоразностных сигналов в первой и второй строках соответственно. В формате H.264 используется структура 4:2:0, что позволяет сократить горизонтальное и вертикальное цветовые разрешения вдвое, но при этом сохранить яркостное разрешение исходным.

1.3. Структура кодека

Видеокодек представляет входную видеопоследовательность (или набор изображений) в сжатом виде и декодирует её таким образом, чтобы создать точную или приближенную копию входной последовательности. В случае, если копия получается идентичной оригиналу, такое кодирование называется кодированием без потерь (loseless), иначе — кодированием с потерями (lossy) [2].

Видеокодек представляет исходную последовательность в виде модели — эффективно закодированного представления, которое затем может использоваться для восстановления приближенной копии исходных данных. В идеальных условиях модель должна представлять исходную последовательность как можно меньшим количеством бит, но при этом обеспечивать как можно большую точность.

Кодировщик видео состоит из трёх основных функциональных блоков — временной модели, пространственной модели и энтропийного кодировщика. На вход временной модели подаются несжатые видеоданные или набор кадров. Временная модель пытается сократить временную избыточность, используя повторяющиеся участки в соседних кадрах. Обычно это делается путём прогнозирования кадра из предыдущего. В H.264 прогноз формируется из одного или нескольких предыдущих или следующих кадров и затем улучшается путём компенсации разницы между кадрами (так называемый прогноз с компенсацией движения). На выход временной модели попадают кадр разницы, получаемый путём вычитания прогноза из текущего кадра, и набор параметров — векторов движения, описывающих то, как именно было компенсировано движение.

Полученный кадр разницы затем поступает на вход пространственной модели, которая анализирует схожесть соседних областей в кадре и использует её для уменьшения пространственной избыточности. В H.264 для этих целей используется трансформация кадров разницы и последующее квантование полученных значений. В процессе трансформации сэмплы преобразуются и представляются в виде набора коэффициентов трансформации, которые затем подвергаются квантованию, что позволяет оставить только малое число наиболее значимых коэффициентов и представить кадр разницы в более компактном виде. В результате на выход пространственной модели попадает набор коэффициентов трансформации.

Полученные параметры временной модели (вектора движения) и параметры пространственной модели (коэффициенты трансформации) затем сжимаются энтропийным кодировщиком. Задача энтропийного кодировщика — устранить статистическую избыточность в полученных на вход данных. Например, энтропийное кодирование может заменить наиболее часто встречающиеся вектора движения на короткие двоичные коды. На выход энтропийного кодировщика поступает сжатый двоичный поток, который затем можно сохранить в файл или передать по сети. Сжатая последовательность состоит из закодированных параметров векторов движения, закодированных коэффициентов трансформации и заголовка.

Декодер видео восстанавливает кадры из сжатого потока данных. Коэффициенты трансформации и вектора движения декодируются энтропийным декодером, после чего пространственная модель восстанавливается до кадра разности. Декодер использует параметры векторов движения в связке с одним или более ранее декодированными кадрами, создавая прогноз для текущего кадра. Итоговый кадр реконструируется путём добавления кадра разности к прогнозу.

Схематично структура кодкека представлена на рисунке 1.

Рис. 1. Структура кодека H.264

1.4. Временная модель

Макроблок, представляющий собой участок кадра размером 16×16 пикселей, является базовой единицей для прогнозирования с компенсацией движения в целом ряде стандартов кодирования видео, включая MPEG-1, MPEG-2, H.261, H.263 и H.264. Для входных данных в формате 4:2:0 макроблок размером 16×16 пикселей представляет собой совокупность четырёх блоков яркости размером 8×8 каждый, двух блоков цветности размером 8×8 для синей и красной цветоразностных компонент соответственно. Кодек H.264 разбивает каждый кадр на макроблоки и затем оперирует ими по отдельности. Для каждого макроблока размером 16×16 выполняются следующие действия.

1. Из ранее закодированного и переданного кадра берутся блоки размером $M \times N$ и выполняется поиск похожих блоков в текущем кадре, для этого блок из предыдущего кадра сравнивается со всеми возможными блоками такого же размера в текущем кадре и выбирается наиболее похожий. Операция сравнения вычитает блоки-кандидаты из исходного блока и получает блоки разницы. Блок-кандидат, которому соответствует блок разницы, содержащий наименьшее количество данных, считается наилучшим совпадением. Процесс поиска наилучшего совпадения называется оценкой движения (motion estimation).
2. Найденное наилучшее совпадение становится прогнозом для текущего блока $M \times N$ и далее используется соответствующий ему блок разницы. Формирование блока разницы путём вычитания прогнозируемого блока из текущего блока называется компенсацией движения (motion compensation).
3. Полученный блок разницы кодируется и передаётся вместе со значениями смещения прогнозируемого блока относительно текущего блока.

Декодер, в свою очередь, получает на вход вектор движения для вычисления прогнозируемой области, декодирует блок разности, складывает его с блоком-предшественником и получает версию текущего блока.

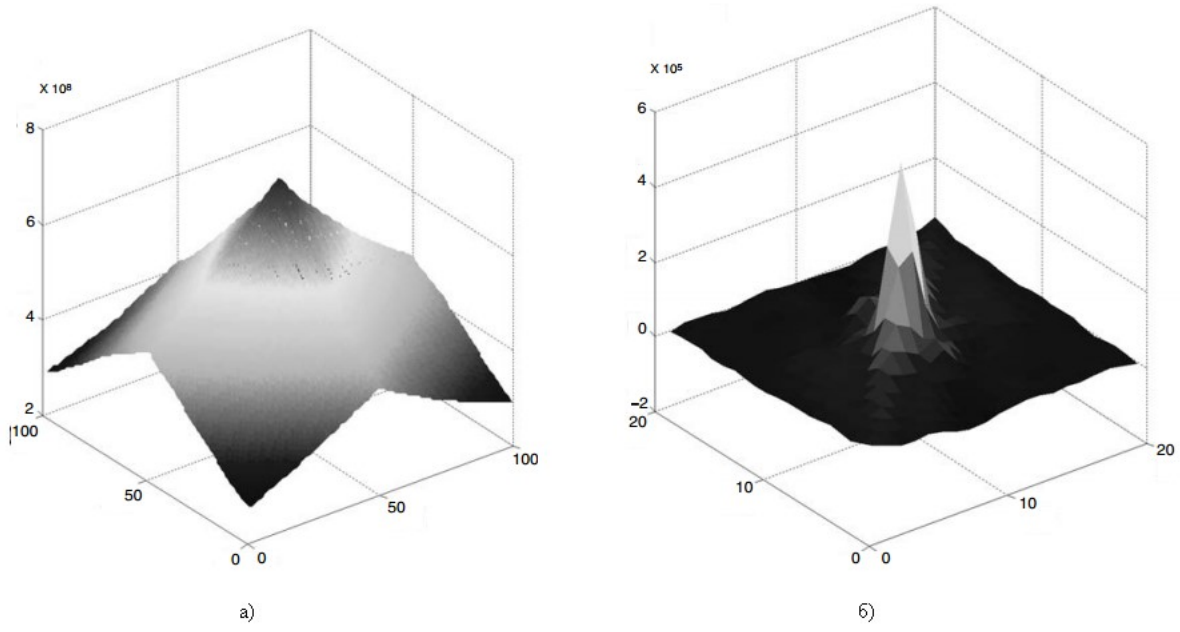
В случаях, когда текущий кадр отличается от предыдущего достаточно сильно, может быть более логичным закодировать макроблок без компенсации движения. В результате кодировщик для каждого макроблока может выбирать между внутрирегиональным режимом (intra, кодирование без компенсации движений) и межрегиональным режимом (inter, кодирование с компенсацией движений).

Движущиеся объекты редко соответствуют границам блоков 16×16, и для достижения лучших показателей более удобным было бы использовать блоки разных размеров. Объекты могут менять своё положение не на целое число пикселей, поэтому при поиске региона-кандидата можно использовать не только регионы с целочисленными координатами, но и субпиксельные.

При уменьшении размеров блоков разности (16×16, 8×8, 4×4) улучшается качество изображения, но увеличиваются сложность (необходимо выполнить большее число операций для поиска блока-кандидата) и количество данных (количество векторов движения, которые необходимо передать). Компромиссом является выбор размера блока в зависимости от характеристик изображения — блоки большего размера для однотонных регионов и меньшие размеры блоков для регионов с большим количеством деталей. H.264 использует блоки варьирующихся размеров для компенсации движения.

Для поиска наиболее точного совпадения в субпиксельных координатах кодек H.264 может сперва найти наиболее точное совпадение в целочисленных координатах, затем выполнить поиск рядом с этим регионом в полупиксельных координатах, а затем и в четвертьпиксельных, если это позволит увеличить точность результата.

1.5. Модель изображения



а) б)

Рис. 2. Двумерные автокорреляционные функции изображения и разницы

На рисунке 2а изображён график двумерной автокорреляционной функции для видеоизображения, на котором высота каждой точки выражает разницу между оригинальным изображением и его пространственно-смещённой копией.

Пик в центре графика соответствует нулевому смещению изображения. С удалением пространственно-смещённой копии от центра оригинального изображения в любом направлении значение функции постепенно уменьшается, образуя на графике постепенный наклон, что свидетельствует о высокой корреляции соседних значений.

На 2б изображён график двумерной автокорреляционной функции для разностного видеоизображения, значения функции на котором стремительно уменьшаются с увеличением смещения, что свидетельствует о низкой корреляции соседних значений. При компенсации движений, уменьшающей корреляцию соседних значений в разнице, сжатие разницы происходит быстрее и эффективнее, чем сжатие исходного видеоизображения. Функция модели изображения состоит в декорреляции изображения или вычета и преобразовании его в форму, которая затем может быть легко сжата энтропийным кодировщиком. Модель изображения состоит из 3 частей — трансформации (декоррелирует и уменьшает размер данных), квантования (уменьшает точность трансформированных данных) и переупорядочивания (выстраивает данные, группируя значимые данные вместе).

1.6. Кодирование с предсказанием

Компенсация движения является частным случаем кодирования с предсказанием, при котором кодировщик формирует предсказание для региона в текущем кадре на основе данных из предыдущих кадров и вычитает прогноз из текущего региона для формирования разницы. Если предсказание успешно, то объём данных, занимаемых разницей, получается меньше объёма фрейма и разница представляется меньшим количеством бит. Подобным же образом участок изображения может быть представлен с помощью других, закодированных ранее, участков в этом же кадре. Кодирование с предсказанием используется для внутрикадрового кодирования в H.264.

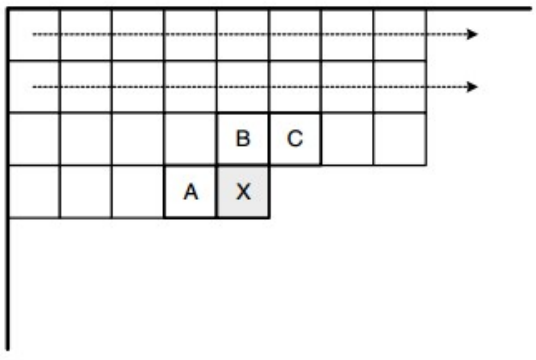


Рис. 3. Пространственное предсказание

Рисунок 3 показывает, как кодируется изображение при пространственном предсказании. Буквой X обозначен пиксель, который необходимо зако-

дировать в данный момент, A, B и C — соседние пиксели, закодированные/декодированные ранее. Кодировщик вычисляет прогноз для X, основываясь на ранее закодированных пикселях A, B и C, вычитает предсказание из X и возвращает разницу. Декодер в свою очередь тоже вычисляет прогноз на основе ранее декодированных пикселях A, B и C и складывает полученное предсказание и разницу, получая на выходе исходный пиксель X.

1.7. Кодирование с преобразованием

Целью шага кодирования с преобразованием является представление изображения или разницы в ином виде (домене преобразования). Выбор метода трансформации может зависеть от ряда критериев.

1. Данные в домене преобразования не должны коррелировать (разделены на части со слабой внутренней связностью) и быть компактными (большая часть преобразованных данных должна содержаться в малом количестве значений).
2. Преобразование должно быть обратимым.
3. Преобразование должно быть как можно менее ресурсоёмким.

В случае с H.264 эту роль могут выполнять дискретное косинусное преобразование или дискретное вейвлет-преобразование. Дискретное косинусное преобразование лучше подходит для использования при компенсации движения не основе блоков, так как само основано на блоках, в отличие от дискретного вейвлет-преобразования, которое предназначено для обработки изображений целиком.

1.8. Дискретное косинусное преобразование

Дискретное косинусное преобразование оперирует блоками изображения или блоками вычетов X размера $N \times N$ и подаёт на выход блоки коэффициентов Y такого же размера. Преобразование X в Y осуществляется по формуле

$$Y = AXA^T.$$

Обратное преобразование выполняется по формуле

$$X = A^T Y A,$$

где X — матрица исходных значений, Y — матрица коэффициентов и A — матрица преобразования размером $N \times N$, элементы которой рассчитываются по формуле:

$$A_{ij} = C_i cos\frac{(2j+1)i\pi}{2N}, C_i = \begin{cases} \sqrt{\frac{1}{N}}, \text{если } i = 0; \\ \sqrt{\frac{2}{N}}, \text{если } i > 0 \end{cases}$$

В результате дискретного косинусного преобразования N^2 значений из исходной матрицы преобразуются в N^2 коэффициентов. Затем можно сохранить

только наиболее значимые коэффициенты, а наименее значимые коэффициенты отбросить. Изменяя количество хранимых коэффициентов, можно управлять качеством итогового видео.

1.9. Квантование

Квантователь преобразует сигнал из множества значений X в множество значений Y. Квантование позволяет представить исходный сигнал в виде меньшего числа бит, так как диапазон выходных значений меньше диапазона входных. Скалярный квантователь преобразует единицу входного сигнала в единицу выходного, а векторный квантователь преобразует группу входных сигналов в группу выходных.

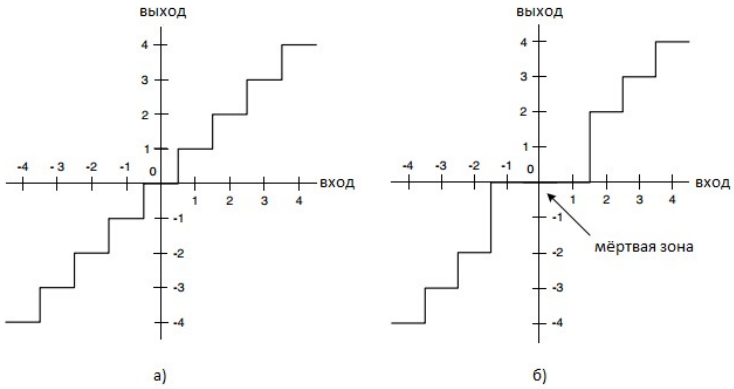


Рис. 4. Скалярные квантователи: а) линейный и б) нелинейный с мёртвой зоной

Рисунок 4 демонстрирует два частных случая скалярного квантователя — линейный квантователь (с прямым соотношением входных и выходных значений) и нелинейный квантователь с мёртвой зоной в области 0 (при котором значения, близкие к 0, переходят в значение 0).

В видеокодеках квантователь обычно состоит из двух частей — прямого квантователя (FQ) в кодировщике и обратного квантователя (IQ) в декодере. Критичным параметром является размер шага (QP). Больший размер шага позволяет уменьшить диапазон квантованных значений, а значит, и уменьшить объём передаваемых данных. Но это повлечёт за собой ухудшение качества, так как восстановленные декодером (обратным квантователем) значения будут является грубым приближением. В случае с меньшим размером шага квантованные данные будут более приближенными к оригиналу, но увеличат объём передаваемых данных и негативно повлияют на эффективность сжатия.

В H.264 квантование используется для отсечения околонулевых малозначимых коэффициентов, полученных в результате дискретного косинусного преобразования. На вход прямого квантователя поступают наборы коэффициентов ДКП, а на выход — разрежённый массив квантованных коэффициентов, большая часть значений в котором представлена нулями.

Векторный квантователь сопоставляет последовательность (вектор) входных данных (такую, как блок изображения) в виде одного значения (кодового слова), а декодер в свою очередь сопоставляет каждое кодовое слово приближенному значению оригинальной входной последовательности. В общем случае процесс векторного квантования состоит из следующих шагов:

1) входное изображение разбивается на блоки размером $M \times N$;
2) из кодовой книги выбирается вектор, наиболее похожий на текущий;
3) на выход передаётся индекс найденного кодового слова;
4) декодер восстанавливает приближенную копию исходного вектора (и, соответственно, исходного блока), используя кодовую книгу и индекс вектора.

Основными проблемами в данной ситуации являются составление кодовой книги таким образом, чтобы обеспечить наилучшее качество восстановленного изображения, и скорость поиска наиболее подходящего вектора в кодовой книге.

1.10. Переупорядочивание

Квантованные коэффициенты трансформации должны быть максимально сжаты перед дальнейшим хранением и передачей. Так как данные, полученные после выполнения квантования, содержат в себе большое число нулевых и малое число ненулевых значений, то имеет смысл произвести переупорядочивание квантованных коэффициентов (сгруппировать ненулевые коэффициенты вместе) и представить нулевые коэффициенты в более компактном виде. Эти шаги необходимо сделать перед выполнением энтропийного кодирования.

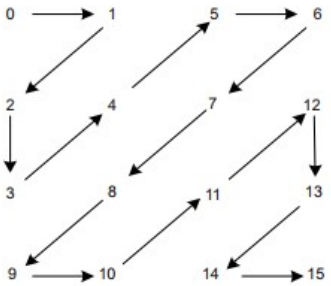


Рис. 5. Сканирование зигзагом

Оптимальный путь для переупорядочивания зависит от распределения ненулевых коэффициентов дискретного косиснусного преобразования ДКП. Для типичного блока кадра путь переупорядочивания будет представлять собой зигзаг (рисунок 5), начинающийся из левого верхнего пикселя. Все коэффициенты ДКП, следуя по пути переупорядочивания, помещаются в одномерный массив. В результате переупорядочивания на выход поступает массив, обычно состоящий из 1 или более кластеров ненулевых коэффициентов в начале,

за которыми следует последовательность нулей. Последовательность ненулевых коэффициентов может быть представлена двумя способами — двумерным кодированием пробег-уровень или трёхмерным кодированием пробег-уровень. В первом случае каждый ненулевой коэффициент представляется в виде пары чисел (пробег, уровень), где пробег — это количество нулей, предшествующих ненулевому значению, а уровень — непосредственное значение коэффициента. Для обозначения того, что ненулевые коэффициенты закончились и далее следуют только нули, в выходную последовательность вставляется специальный символ, обозначающий конец последовательности. Во втором случае каждый элемент последовательности кодируется тремя значениями (пробег, уровень, последний), где пробег и уровень соответствуют аналогичным значениям в первом случае, а последний — признак конца последовательности (0, если далее следует ещё один ненулевой коэффициент, и 1, если данный ненулевой коэффициент — последний).

1.11. Энтропийный кодировщик

Энтропийное кодирование преобразует набор значений, представляющих элементы видеопоследовательности, в сжатый поток бит, подходящий для хранений или передачи. Наборы значений могут включать в себя квантованные коэффициенты трансформации, вектора движения, заголовки макроблоков и изображений и дополнительную информацию.

Некоторые элементы видеопоследовательности могут довольно сильно коррелировать друг с другом, например, вектора движения или коэффициенты ДКП в соседних блоках. Эффективность сжатия может быть улучшена, если кодировать не отдельный блок или макроблок, а разницу, зависимую от реальных и ранее закодированных данных.

Вектор движения блока или макроблока означает смещение относительно связанного блока из предыдущего кадра. Вектора соседних блоков довольно часто коррелируют, так как движущиеся объекты могут занимать большие участки кадра. Это утверждение наиболее справедливо для блоков малых размеров (4×4) и больших движущихся объектов. Сжатие векторов движения может быть выполнено путём вычисления зависимости вектора от ранее закодированных векторов. Разница между реальным вектором и спрогнозированным значением затем кодируется и передаётся вместо оригинального значения.

Кодировщик с переменной длиной преобразует набор входных значений в набор кодовых слов переменной длины. При этом часто встречающиеся значения представляются в виде кодовых слов меньшей длины. Кодирование Хаффмана сопоставляет каждое входное значение с кодовым словом переменной длины, основываясь на частоте вхождения разных входных значений. Согласно оригинальной схеме, предложенной Хаффманом в 1952 году, перед кодированием необходимо сперва рассчитать частоту вхождений всех входных значений и построить набор кодов переменной длины. Если вероятностное распределение рассчитано верно, код Хаффмана позволяет представить входные данные в сравнительно более компактной форме.

Код Хаффмана имеет два недостатка при его практическом применении в работе видеокодека. Во-первых, декодер должен использовать тот же набор кодовых слов, что использовал кодировщик. Передача таблицы соответствия кодов Хаффмана исходным данным требует дополнительных расходов и снижает эффективность сжатия, особенно для видеопоследовательностей малой длины. Во-вторых, таблица вероятностей для видеопоследовательности большой длины не может быть построена до тех пор, пока не закодирован весь видеопоток, что может повлечь за собой большие задержки в процессе кодирования. Для решения этой проблемы H.264 предлагает заранее рассчитанные таблицы вероятностей, созданные на основе некоторого стандартного видеоматериала.

Стандарт MPEG-4 позволяет представить трёхмерные коды квантованных коэффициентов (пробег, уровень, последний) как одно из 102 кодовых слов переменной длины, таблица которых определена в стандарте. В случае, если кодового слова для нужного трёхмерного кода нет в таблице, то трёхмерный код представляется в виде 13-битного кода, содержащего все три значения, и записывается после управляющей последовательности (0000011). Векторы движения представляются парой кодовых слов переменной длины (одно значение для координаты x и одно — для y). Стандарт MPEG-4 также предлагает для этих целей готовую таблицу соответствия.

2. Исследование способов сокрытия данных в H.264

2.1. Сокрытие данных в векторах движения

Одним из наиболее очевидных способов сокрытия данных в видеопотоке является сокрытие в векторах движения [3,4]. Для уменьшения влияния скрытых данных на итоговое видео данные записываются только в младший бит обеих координат вектора. Качество итогового видео тем лучше, чем большая точность вектора используется при кодировании. Полу- или четвертьпиксельная точность вектора движения позволяет скрывать данные практически без ущерба для итогового изображения.

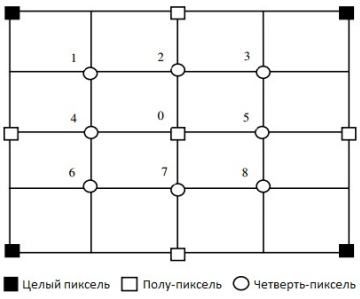


Рис. 6. Нумерация пикселей

Частным случаем является использование свойства чётности координат вектора для сокрытия данных. При таком подходе для хранения координат вектора

используется четвертьпиксельная точность вектора (рисунок 6), а все возможные точки поиска разбиваются на 2 группы — (0, 1, 3, 6, 8) и (2, 4, 5, 7). Каждой из двух групп точек ставится в соответствие бит 0 или 1. Сокрытие данных происходит в процессе оценки движения. После поиска наилучшего полупиксельного совпадения все соседние четвертьпиксели разбиваются на группы и в каждой группе выбирается наилучший кандидат. Затем выбирается группа, соответствующая значению, которое необходимо скрыть (0 или 1). Наилучший кандидат из соответствующей группы считается итоговым наилучшим кандидатом.

2.2. Сокрытие данных в макроблоках

Ещё одним методом является сокрытие данных в макроблоках [5], а точнее, в признаке принадлежности макроблоков к определённым группам слайсов. Кадр в H.264 состоит из макроблоков, которые затем могут быть разбиты на слайсы, а слайсы объединены в группы слайсов. Каждая группа слайсов состоит из одного или более слайса и макроблоков, расположенных в любом порядке. Соответствие макроблоков группам слайсов может использоваться для сокрытия данных в видеопотоке. Так как макроблоки могут быть произвольным образом сопоставлены с группой слайсов, то биты данных могут быть скрыты в поле идентификатора группы слайсов в макроблоке.

В случае, если существуют две группы слайсов, назначение макроблока в первую группу слайсов может кодировать бит 0, а назначение во вторую — бит 1. Таким образом, каждый макроблок может скрывать 1 бит информации. Так как стандарт H.264 позволяет использовать максимум 8 групп слайсов на кадр, количество кодируемых значений можно увеличить до 8, тем самым позволяя в одном макроблоке скрывать до 3 бит информации (таблица 1).

Таблица 1. Соответствие числа групп слайсов количеству возможных скрытых бит

Количество групп слайсов в кадре	Возможные скрытые значения в макроблоке	Скрытых бит в макроблоке
2	0; 1	1
4	00; 01; 10; 11	2
8	000; 001; 010; 011; 100; 101; 110; 111	3

Если кадр содержит m макроблоков, то в кадре можно скрыть $m \times n$ бит информации. Процесс кодирования выполняется следующим образом. В первую очередь считываются $n \times m$ бит скрываемых данных, где n — значение 1, 2 или 3, согласно таблице 1. Затем каждому макроблоку в кадре по очереди ставится в соответствие нужная группа слайсов в зависимости от прочитанных скрываемых данных. Процесс декодирования выглядит соответствующим образом — для каждого макроблока в кадре берётся значение поля идентификатора группы слайсов, значение этого поля является частью скрытого сообщения.

2.3. Сокрытие данных в квантованных коэффициентах ДКП

Также существует способ сокрытия данных в коэффициентах, полученных в результате дискретного косинусного преобразования и последующего квантования [6]. При данном подходе сначала выбирается некое граничное значение от 0 до 15, определяющее максимально допустимый номер коэффициента ДКП, разрешённого для модификации. Выбор граничного значения зависит от количества информации, которую необходимо скрыть. Кодирование осуществляется путём перебора всех блоков разницы и выбора из них таких блоков, для которых последний ненулевой коэффициент имеет индекс не меньше, чем граничное значение. Если подходящий коэффициент (V_a) найден, то он изменяется в зависимости от того, какое значение (D_a) необходимо скрыть:

Если $D_a = 1$:

$$D_a = \begin{cases} V_a, \text{если } V_a\%2 = 1; \\ V_a - 1, \text{если } V_a\%2 = 0 \text{ и } V_a > 0; \\ V_a + 1, \text{если } V_a\%2 = 0 \text{ и } V_a < 0. \end{cases}$$

Если $D_a = 0$:

$$D_a = \begin{cases} V_a, \text{если } V_a\%2 = 0; \\ V_a - 1, \text{если } V_a\%2 = 1 \text{ и } V_a > 0; \\ V_a + 1, \text{если } V_a\%2 = 1 \text{ и } V_a < 0. \end{cases}$$

То есть если встраиваемое значение — 1, то значение коэффициента будет нечётным, а если 0 — то чётным.

Процесс декодирования выглядит следующим образом — сначала из всех наборов коэффициентов выбираются те, у которых индекс последнего ненулевого коэффициента не меньше, чем граничное значение (аналогично процессу кодирования). Из каждого подходящего набора коэффициентов выбирается последний ненулевой (V_a) и выполняется извлечение скрытого бита данных (D_a) по следующему принципу:

$$D_a = \begin{cases} 1, \text{если } V_a\%2 = 1; \\ 0, \text{если } V_a\%2 = 0. \end{cases}$$

Преимуществом такого подхода является простота сокрытия данных. Так как данный алгоритм использует признак чётности для сокрытия данных и вероятности того, что значение последнего ненулевого коэффициента, удовлетворяющего условию, будет чётным или нечётным, равны и составляют 50%, то есть 50-процентная вероятность того, что коэффициент останется неизменным после сокрытия данных. Это положительно сказывается как на качестве результирующего видео, так и на скорости кодирования. Скорость кодирования видео при использовании этого метода практически не изменяется, а декодирование данных происходит без каких-либо потерь.

3. Экспериментальное исследование возможности сокрытия данных

3.1. Подготовка эксперимента

Из рассмотренных методов сокрытия информации было решено выбрать сокрытие в коэффициентах дискретного косинусного преобразования. Так как реализация собственного кодировщика и декодировщика является очень сложной задачей, было решено модифицировать уже готовое решение. Большинство решений не предоставляют свои исходные коды для модификации и не позволяют вмешиваться в процесс своей работы.

Были найдены две открытые реализации кодека H.264 — одна от Института Генриха Герца [7], а вторая от компании Cisco [8]. Выбор пал на вторую как наиболее надёжную, активно разрабатываемую и поддерживаемую (последняя стабильная версия опубликована 11 июля 2016). Данный кодек написан на языке C++ с ассемблерными вставками.

Кодировщик в качестве параметра командной строки принимает путь к файлу конфигурации, в котором описываются все настраиваемые параметры, такие как количество кадров в секунду и размер кадра, а также пути к входному и выходному файлам. Входной файл должен представлять собой набор сырых данных в формате YUV ($YCbCr$), а в выходной файл будет сохранено видео в формате H.264 без контейнера.

Декодер в качестве параметров командной строки позволяет указать пути к входному и выходному файлам. Здесь в качестве входного файла должен выступать файл с H.264 видео без контейнера, а в качестве выходного будет создан файл с сырыми YUV данными. Для получения сырого H.264 видео из файла с видео, использующего контейнер MPEG-4, можно использовать утилиту ffmpeg [9–11].

3.2. Модификация кодека

Модификация кодека будет заключаться в добавлении программного кода, позволяющего считывать встраиваемые данные из файла, встраивающего данные в видеофайл, извлекающего данные из видеофайла и записывающего извлечённые данные в файл.

Для начала необходимо реализовать методы, осуществляющие чтение данных из файла и возвращающие значение следующего бита данных.

1. В файл walsenc.cpp добавим метод *ReadPayloadFile* для чтения файла с данными.
2. В этом же файле в структуру *SFilesSet* добавим строковую переменную *strPlFile*, а в метод *ParseConfig* добавим чтение параметра из файла конфигурации.
3. Добавим новый объект типа *FILE** с именем *pFpPl* и код открытия файла на чтение и соответствующие проверки.

4. Добавим вызов метода *ReadPayloadFile* и возвращенное им значение передадим в вызов метода *EncodeFrame* в качестве второго аргумента.
5. Изменим вызов и объявление метода *WelsEncRecI16x16Y*, добавляя массив со скрываемыми данными в качестве последнего аргумента.
6. В самый конец метода *WelsEncRecI16x16Y* добавим логику, непосредственно связанную с сокрытием информации.

Теперь необходимо реализовать алгоритм восстановления данных в декодере.

1. Сначала необходимо модифицировать разбор параметров командной строки. Для этого в метод *main* файла h264dec.cpp добавим строковую переменную *strPayloadFile* и модифицируем код так, чтобы значение переменной бралось из третьего аргумента командной строки.
2. Модифицируем вызов и описание метода *H264DecodeInstance*, добавив путь к файлу с восстановленными данными в качестве последнего параметра.
3. В метод *H264DecodeInstance* добавим код, осуществляющий открытие файла для хранения восстановленных данных на запись, создание массива для хранения декодированных данных и запись получившихся данных в файл.
4. Изменим объявление и вызов метода *DecodeFrameNoDelay*, добавив в качестве последнего аргумента параметр типа *char** для хранения восстановленных данных.
5. Восстановление данных можно выполнить внутри метода *RecI16x16Mb*, изменив его объявление и вызов, добавив в качестве последнего аргумента файл восстановленных данных, как и в предыдущем случае.
6. В самый конец метода *RecI16x16Mb* добавим логику, непосредственно ответственную за восстановление данных.

3.3. Проверка корректности работы

Для проверки корректности работы необходимо выполнить следующие шаги.

1. Перекомпилируем кодек.
2. Создадим файл с данными, которые необходимо скрыть:
   echo “Data hiding in H.264 video. –OmSU, 2017” > payload.txt
3. Создадим файл конфигурации. В качестве основы можно взять файл welsenc_vd_1d.cfg, поставляющийся с кодеком, указать в нём следующие параметры:
   - InputFile — имя входного файла (source.yuv);
   - OutputFile — имя выходного файла (test.h264);
   - PayloadFile — имя файла со скрываемыми данными (payload.txt);
   - SourceWidth — ширина изображения в исходном видео (1920);
   - SourceHeight — высота изображения в исходном видео (1080).
4. Запустим кодировщик, передав ему файл конфигурации в качестве единственного параметра:

h264enc payload.cfg

5. Запустим декодер со следующими параметрами:
   ../h264dec test.h264 test.yuv payload.new
6. Сравним исходные и полученные данные.

Результат эксперимента показал, что восстановленный файл содержит все данные из оригинального файла, что свидетельствует о правильной работе кодека и реализованного алгоритма.

Заключение

В данной работе были рассмотрены основные способы сокрытия информации в видеопотоке H.264. Для проверки возможности встраивания информации в видеопоток был выбран метод встраивания в коэффициенты ДКП. На основе кодека OpenH264 была реализована возможность встраивать и извлекать информацию. Результаты эксперимента показали, что выбраная схема применима для решения поставленной задачи.

Литература

1. Information technology – Coding of audio-visual objects – Part 14: MP4 file format. International Standard, 2003.
2. Richardson I.E.G. H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia. Wiley, 2003.
3. Li Y., Chen H.-X., Zhao Y. A new method of data hiding based on H.264 encoded video sequences // IEEE 10th International Conference on Signal Processing (ICSP). 2010. P. 1833–1836.
4. Aly H.A. Data hiding in motion vectors of compressed video based on their associated prediction error // IEEE Trans. Inform. Forensics Security. 2011. V. 6, № 1. P. 14–18.
5. Lathikanandini M., Suresh J. Steganography in MPEG Video Files using MACROBLOCKS // IEEE Trans. Inform. Forensics Security. 2012. V. 3, № 1.
6. Wong K., Tanaka K., Takagi K., Nakajima Y. Complete video quality-preserving data hiding // IEEE Transactions on Circuits and Systems for Video Technology. 2009. V. 19, № 10.
7. H.264/MPEG4-AVC URL: `https://www.hhi.fraunhofer.de/index.php?id=468&L=1` (дата обращения: 15.03.2017).
8. OpenH264 URL: `https://www.openh264.org/` (дата обращения: 15.03.2017).
9. H.264/MPEG-4 AVC Video Compression Tutorial. URL: `http://web.cs.ucla.edu/classes/fall03/cs218/paper/H.264\_MPEG4\_Tutorial.pdf` (дата обращения: 23.12.2016).
10. Extracting h264 raw video stream from mp4 or flv with ffmpeg generate an invalid stream. URL: `http://stackoverflow.com/questions/19300350/extracting-h264-raw-video-stream-from-mp4-or-flv-with-ffmpeg-generate-an-invalid` (дата обращения: 23.12.2016).
11. How to convert raw H.264 stream from video call to mp4 file. URL: `http://stackoverflow.com/questions/29293836/`

how-to-convert-raw-h-264-stream-from-video-call-to-mp4-file (дата обращения: 23.12.2016).

INVESTIGATION OF METHODS OF INFORMATION HIDING IN THE MPEG VIDEO STREAM

D.M. Brechka
Ph.D.(Eng.), Associate Professor, e-mail: dbrechkawork@yandex.ru
A.A. Litvinenko
Student, e-mail: a.litvinenko@outlook.com

Dostoevsky Omsk State University

Abstract. The article discusses the methods of information hiding in the MPEG-4 video stream using the H.264 codec. To test the possibility of embedding of information in the video stream, there was chosen method of embedding in the coefficients of the discrete cosine transform. The ability of embedding and retrieving of information was tested on modified OpenH264 codec.

Keywords: codec, MPEG-4, H.264, embedding data, video stream, hiding information.

Дата поступления в редакцию: 17.03.2017