(19) Евразийское патентное ведомство

(11) 044215 (13) B1

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ЕВРАЗИЙСКОМУ ПАТЕНТУ

(45) Дата публикации и выдачи патента
2023.07.31

(21) Номер заявки
202190284

(22) Дата подачи заявки
2019.08.01

(51) Int. Cl. ***H04N 19/33*** (2014.01)
H04N 19/91 (2014.01)
H04N 19/93 (2014.01)

(54) ЭНТРОПИЙНОЕ КОДИРОВАНИЕ УСИЛЕНИЯ СИГНАЛА

(31) **1812708.4; 1812709.2; 1812710.0; 1903844.7; 1904014.6; 1904492.4; 1905325.5**
(32) **2018.08.03; 2018.08.03; 2018.08.03; 2019.03.20; 2019.03.23; 2019.03.29; 2019.04.15**
(33) **GB**
(43) **2021.05.18**
(86) **PCT/GB2019/052166**
(87) **WO 2020/025964 2020.02.06**

(71)(73) Заявитель и патентовладелец:
B-НОВА ИНТЕРНЭШНЛ ЛТД (GB)

(72) Изобретатель:
Меарди Гвидо (GB)

(74) Представитель:
Нилова М.И. (RU)

(56) ANONYMOUS: "H.264 and MPEG-4 Video Compression, chapter 6, "H.264/MPEG4 Part 10", Iain E. Richardson", NOT KNOWN, 17 October 2003 (2003-10-17), XP030001626, pages 159-161; figures 6.1,6.2 page 198 page 201

US-A1-2009097548

"Working Draft of Low Complexity Enhancement Video Coding", 126. MPEG MEETING; 20190325 - 20190329; GENEVA; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11), no. n18454 18 April 2019 (2019-04-18), XP030208724, Retrieved from the Internet:URL:http://phenix.int-evry.fr/mpeg/doc_end_user/documents/126_Geneva/wgll/wl8454.zip N18454.docx [retrieved on 2019-09-26] page 28 pages 45-51

(57) Представлен способ кодирования видеосигнала, включающий: прием входного кадра; обработку входного кадра для генерирования по меньшей мере одного набора остаточных данных, позволяющих декодеру восстановить входной кадр из опорного кадра восстановления; и применение операции кодирования длин серий к набору остаточных данных, при этом операция кодирования длин серий включает генерирование байтового потока, закодированного по длинам серий, содержащего набор символов, представляющих ненулевые значения данных из набора остаточных данных, и подсчеты последовательных нулевых значений набора остаточных данных. В некоторых вариантах воплощения способ включает применение операции кодирования Хаффмана к набору символов. Также предоставляется способ декодирования, а также устройства и машиночитаемый носитель.

044215 B1

044215 B1

Уровень техники

Ранее была предложена гибридная технология кодирования с обратной совместимостью, например, в WO 2014/170819 и WO 2018/046940, содержание которых включено в настоящий документ путем отсылки.

Предложен способ, который предусматривает: разбор потока данных на первые порции кодированных данных и вторые порции кодированных данных; реализует первый декодер для декодирования первых порций кодированных данных при первой передаче сигнала; реализует второй декодер для декодирования вторых порций кодированных данных в данные восстановления, данные восстановления, определяющие, как модифицировать первую передачу сигнала; и применяет данные восстановления к первой передаче сигнала для получения второй передачи сигнала.

В нем дополнительно предлагается дополнение, в котором набор остаточных элементов может использоваться для восстановления воспроизведения первой временной выборки сигнала. Генерируется набор элементов пространственно-временной корреляции, связанных с первой временной выборкой. Набор элементов пространственно-временной корреляции указывает степень пространственной корреляции между множеством остаточных элементов и степень временной корреляции между первыми опорными данными на основе воспроизведения и вторыми опорными данными на основе воспроизведения второй временной выборки сигнала. Набор элементов пространственно-временной корреляции используется для генерации выходных данных.

Типичные технологии кодирования видео предусматривают применение операции энтропийного кодирования к выходным данным. Существует потребность в простой и быстрой схеме энтропийного кодирования с низкой сложностью для применения сжатия данных к данным восстановления или остаточным элементам вышеупомянутых предложенных технологий или другим подобным остаточным данным.

Раскрытие изобретения

Согласно аспектам изобретения предоставляется способ энтропийного кодирования или декодирования остаточных данных, причем остаточные данные могут использоваться для исправления или расширения данных базового потока, например кадра видео, закодированного с использованием унаследованной технологии кодирования видео.

Согласно аспекту изобретения предоставляется способ кодирования видеосигнала. Способ включает: получение входного кадра; обработку входного кадра для генерирования остаточных данных, остаточные данные формируют часть расширенного потока и позволяют декодеру восстановить входной кадр на основе опорного восстановленного кадра; и применение операции кодирования длин серий к остаточным данным, при этом операция кодирования длин серий включает генерирование байтового потока, закодированного по длинам серий, содержащего набор символов, представляющих ненулевые значения остаточных данных и подсчеты последовательных нулевых значений остаточных данных, причем способ характеризуется следующими этапами: применение канонической операции кодирования Хаффмана к набору символов для генерирования данных, закодированных по Хаффману, содержащих набор кодов, представляющих байтовый поток, закодированный по длине серий; сравнение размера данных по меньшей мере части байтового потока, закодированного по длине серий, с размером данных по меньшей мере соответствующей части данных, закодированных по Хаффману; и вывод байтового потока, закодированного по длине серий, или данных, закодированных по Хаффману, в выходной битовый поток (102, 103) на основе того, какой из них имеет меньший размер данных.

Таким образом, способ кодирования обеспечивает несложное, простое и быстрое решение для сжатия остаточных данных. В решении используются уникальные характеристики остаточных данных, такие как относительное появление нулевых значений, вероятная группировка нулевых значений на основе порядка сканирования процесса преобразования генерирования остаточных данных (если преобразование используется) и относительное разнообразие значений данных, а также их потенциальная относительная частота/редкость в остаточных данных.

Набор символов в кодированном потоке байтов может быть последовательным. Подсчет последовательных нулевых значений также может называться серией нулей. Остаточные данные, над которыми выполняется операция кодирования длин серий, могут представлять квантованные остаточные данные, предпочтительно квантованный набор коэффициентов преобразования. Квантованный набор коэффициентов преобразования может быть упорядочен слоем, то есть по наборам коэффициентов одного типа, по плоскости, по уровню качества или по поверхности. Эти термины дополнительно описаны и определены в данном документе. Символы могут быть последовательными в соответствующем порядке сканирования операции преобразования, так что остаточные данные могут быть легко сопоставлены с опорным восстановленным кадром.

Предпочтительно операция кодирования длин серий включает: кодирование ненулевых значений данных остаточных данных в по меньшей мере первом типе символа; и кодирование подсчетов последовательных нулевых значений во втором типе символа, так что остаточные данные кодируются как последовательность символов разных типов. Таким образом, остаточные данные могут быть закодированы как последовательность символов, содержащая тип значения данных и серию нулей. Типы обеспечивают скорость и простоту декодирования в декодере, но также облегчают дальнейшую операцию энтропийно-

го кодирования, как описано ниже. Структурирование данных в минимизированный набор кодов фиксированной длины разных типов применяется для выполнения последующих шагов.

В некоторых вариантах воплощения операция кодирования длин серий включает следующие действия: кодирование значений данных остаточных данных в первый тип символа и третий тип символа, первый и третий типы символа, каждый из которых содержит часть значения данных, так что части могут быть объединены в декодере для восстановления значения данных. Каждый тип символа может иметь фиксированный размер и может быть байтом. Значения данных, превышающие пороговое значение или превышающие размер, доступный в типе символа, могут быть легко переданы или сохранены с использованием потока байтов. Структурирование данных в байты или символы не только облегчает декодирование, но также облегчает энтропийное кодирование символов фиксированной длины, как будет рассмотрено далее в этом документе. Добавление третьего типа символа позволяет легко различать три типа символов друг от друга на стороне декодера. Там, где мы говорим о типах символов, могут использоваться термины блоки, байты (где символ является байтом), контексты или типы.

Операция кодирования длин серий может включать следующие действия: сравнение размера каждого значения данных остаточных данных, которые должны быть кодированы, с пороговым значением; и кодирование каждого значения данных в символ первого типа символа, если размер ниже порогового, и кодирование части каждого значения данных в символ первого типа символа и часть каждого значения данных в символ третьего типа символа, если размер выше порогового значения.

Если размер превышает пороговое значение, способ может содержать установку флага в символе первого типа символа, указывающего, что часть представленного значения данных кодируется в дополнительный символ третьего типа символа. Флаг может быть флагом переполнения или битом переполнения и может в некоторых примерах быть младшим значащим битом символа или байта. Если наименее значимый бит символа является флагом, значение данных или часть значения данных может содержаться в оставшихся битах байта. Установка бита переполнения как младшего значащего бита облегчает комбинирование с последующими символами.

Способ может дополнительно включать вставку флага в каждый символ, указывающего тип символа, кодируемого следующим в байтовом потоке, закодированном по длине серий. Флаг может быть битом переполнения, как описано выше, или может дополнительно быть флагом выполнения или битом выполнения, указывающим, содержит ли следующий символ серию нулей, или значение данных, или часть значения данных. Если флаг представляет, содержит ли следующий символ серию (или подсчет последовательных нулей), флаг может быть битом символа, предпочтительно самым старшим битом символа. Подсчет последовательных нулей или значение данных может содержаться в оставшихся битах символа. Таким образом, «серия нулей» или символ второго типа содержит 7 доступных битов байта для подсчета, а «данные» или символ первого типа содержат 6 доступных битов для значения данных, где бит переполнения не установлен и 7 доступных битов для значения данных, в котором установлен бит переполнения.

В общем, флаг может быть различным для каждого типа символа и может указывать тип символа, следующего в потоке.

В предпочтительных вариантах воплощения вышеупомянутого аспекта способ дополнительно включает применение дополнительной операции энтропийного кодирования к набору символов, созданных операцией кодирования длин серий. Таким образом, символы фиксированной длины, намеренно созданные структурой операции кодирования длин серий, могут быть преобразованы в коды переменной длины, чтобы уменьшить общий размер данных. Структура кодирования длин серий разработана для того, чтобы создавать символы фиксированной длины с высокой частотой, чтобы облегчить дальнейшее выполнение улучшенной операции энтропийного кодирования и обеспечить общее уменьшение размера данных. Дополнительная операция энтропийного кодирования использует вероятность или частоту появления символа в байтовом потоке, созданном операцией кодирования длин серий.

В предпочтительном примере дополнительная операция энтропийного кодирования представляет собой операцию кодирования Хаффмана или операцию арифметического кодирования. Предпочтительно способ дополнительно включает применение операции кодирования Хаффмана к набору символов для генерации данных, закодированных кодированием Хаффмана, содержащих набор кодов, представляющих байтовый поток, закодированный по длине серии. Операция кодирования Хаффмана принимает в качестве входных символов байтовый поток, закодированный по длине серии, и выводит множество кодов переменной длины в потоке битов. Кодирование Хаффмана представляет собой способ кодирования, который может эффективно сокращать коды фиксированной длины. Структура байтов, закодированных по длине серии, означает, что символы двух типов с большой вероятностью будут часто реплицироваться, а это означает, что для каждого повторяющегося символа потребуется всего несколько битов. По всей плоскости одно и то же значение данных, вероятно, будет реплицировано (особенно, когда остаточные данные квантованы), и, следовательно, код переменной длины может быть небольшим (но повторяться).

Кодирование Хаффмана также хорошо оптимизировано для программной реализации (как ожидается здесь) и является эффективным сточки зрения вычислений. При этом используется минимальный объем памяти, а декодирование выполняется быстрее по сравнению с другими способами энтропийного ко-

дирования, поскольку этапы обработки представляют простое пошаговое перемещение по дереву. Вычислительные преимущества возникают из-за формы и глубины дерева, созданного с помощью специально разработанной операции кодирования длин серий.

Операции кодирования могут выполняться над группой коэффициентов, то есть на слое, на плоскости кадра, на уровне качества или на всей поверхности или кадре. То есть статистика или параметры каждой операции могут быть определены на основе группы значений данных и нулевых значений данных, которые должны быть кодированы, или символов, представляющих эти значения.

Операция кодирования Хаффмана может быть канонической операцией кодирования Хаффмана, так что данные, закодированные Хаффманом, содержат длину кода для каждого уникального символа из набора символов, причем длина кода представляет длину кода, используемого для кодирования соответствующего символа. Каноническое кодирование Хаффмана способствует уменьшению количества параметров кодирования, которые необходимо сообщать между кодером и декодером в метаданных, гарантируя, что идентичные длины кода прикрепляются к символам последовательным образом. Кодовая книга для декодера может быть выведена, и необходимо отправить только длины кодов. Таким образом, канонический кодер Хаффмана особенно эффективен для неглубоких деревьев, где имеется большое количество различных значений данных.

В некоторых примерах способ дополнительно включает сравнение размера данных по меньшей мере части байтового потока, закодированного по длине серий, с размером данных по меньшей мере части данных, закодированных по Хаффману; и вывод байтового потока, закодированного по длине серий, или данных, закодированных по Хаффману, в выходном битовом потоке на основе сравнения. Таким образом, каждый блок или секция входных данных выборочно отправляется для уменьшения общего размера данных, гарантируя, что декодер может легко различать типы кодирования. Разница может находиться в пределах порогового значения или допуска, так что, хотя одна схема может приводить к меньшему количеству данных, вычислительная эффективность может означать, что одна схема является предпочтительной.

Чтобы различать схемы, способ может дополнительно включать добавление флага к метаданным конфигурации, сопровождающим выходной поток битов, указывающего, представляет ли поток битов байтовый поток, закодированный по длине серии, или данные, закодированные по Хаффману. Это быстрый и эффективный способ передачи сигналов. Альтернативно, декодер может идентифицировать используемую схему из структуры данных, например, данные, закодированные по Хаффману, могут иметь часть заголовка и часть данных, в то время как данные, закодированные по длине серии, могут содержать только часть данных, и, таким образом, декодер может уметь различать две схемы.

Операция кодирования Хаффмана может создавать отдельную таблицу частот для символов первого типа символа и символов второго типа символа. Кроме того, для каждого типа символа может применяться отдельная операция кодирования Хаффмана. Эти концепции способствуют особой эффективности схемы кодирования с переменной длиной кода. Например, если одно и то же значение данных реплицируется по плоскости, как это вероятно при кодировании видео, где сцена может иметь похожие цвета или ошибки/улучшения, для этих значений может потребоваться только несколько кодов (где коды не искажаются из-за символа типа "серия нулей"). Таким образом, этот вариант воплощения изобретения особенно выгоден для использования в сочетании с вышеописанной операцией кодирования длин серий. Как отмечалось выше, когда значения квантованы, символы могут быть реплицированы и не могут быть слишком многими разными значениями в одном кадре или группе коэффициентов.

Использование другой частотной таблицы для каждого типа символа обеспечивает особое преимущество в операции кодирования Хаффмана после специально разработанной операции кодирования длин серий.

Таким образом, способ может включать определение типа символа, который должен быть закодирован следующим, выбор таблицы частот на основе типа следующего символа, декодирование следующего символа с использованием выбранной таблицы частот и вывод закодированного символа в последовательности. Таблица частот может соответствовать уникальной кодовой книге.

Способ может включать создание заголовка потока, который может содержать указание множества длин кода, так что декодер может выводить длины кода и соответствующие символы для операции канонического декодирования Хаффмана. Длина кода может быть, например, длиной каждого кода, используемого для кодирования конкретного символа. Следовательно, длина кода имеет соответствующий код и соответствующий символ. Заголовок потока может быть первым типом заголовка потока и дополнительно содержать указание символа, связанного с соответствующей длиной кода из множества длин кода. В качестве альтернативы заголовок потока может быть заголовком потока второго типа, и способ может дополнительно включать упорядочивание множества длин кодов в заголовке потока на основе заранее определенного порядка символов, соответствующих каждой из длин кода, так что длины кода могут быть ассоциированы с соответствующим символом в декодере. Каждый тип обеспечивает эффективную сигнализацию параметров кодирования в зависимости от символов и длин, которые должны быть декодированы. Длины кода могут сигнализироваться как разница между длиной и другой длиной, предпочтительно сигнализируемой минимальной длиной кода.

Второй тип заголовка потока может дополнительно содержать флаг, указывающий, что символ в

заранее определенном порядке возможного набора символов не существует в наборе символов в байтовом потоке, закодированном по длине серий. Таким образом, в заголовке должна содержаться только необходимая длина.

Способ может дополнительно включать сравнение количества уникальных кодов в данных, закодированных по Хаффману, с пороговым значением и генерирование первого типа заголовка потока или второго типа заголовка потока на основе сравнения.

Способ может дополнительно включать сравнение количества ненулевых символов или символов данных с пороговым значением и генерирование первого типа заголовка потока или второго типа заголовка потока на основе сравнения.

В соответствии с дополнительным аспектом изобретения может быть предоставлен способ декодирования видеосигнала, при этом способ включает: извлечение кодированного потока битов; декодирование кодированного битового потока для генерирования остаточных данных, и, восстановление оригинального кадра видеосигнала на основе остаточных данных и опорного восстановленного кадра, в котором этап декодирования кодированного битового потока содержит: применение операции кодирования длин серий, чтобы генерировать остаточные данные, при этом операция кодирования длин серий включает идентификацию набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений остаточных данных; анализ набора символов для получения ненулевых значений данных остаточных данных и подсчета последовательных нулевых значений; и генерирование остаточных данных на основе ненулевых значений данных и подсчеты последовательных нулевых значений, причем способ характеризуется тем, что этап декодирования закодированного битового потока включает: извлечение флага из метаданных конфигурации, сопровождающих закодированный битовый поток (102. 103), указывающего, содержит ли закодированный битовый поток байтовый поток, закодированный по длине серии, или данные, закодированные по Хаффману; и выборочное применение канонической операции кодирования Хаффмана к закодированному битовому потоку (102, 103) на основе флага для генерирования набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений остаточных данных, при этом этап применения операции кодирования длин серий для генерирования остаточных данных выполняют над наборами символов.

Операция кодирования длин серий может содержать: идентификацию символов первого типа символа, представляющего ненулевые значения данных остаточных данных; идентификацию символов второго типа символа, представляющих подсчеты последовательных нулевых значений, так что последовательность символов разных типов декодируется для генерирования остаточных данных; и анализ набора символов согласно соответствующему типу каждого символа.

Операция кодирования длин серий может содержать идентификацию символов первого типа символа и символов третьего типа символа, причем каждый первый и третий типы символа представляют часть значения данных; синтаксический анализ символа первого типа символа и символа третьего типа символа для получения частей значения данных; и объединение производных частей значения данных в значение данных.

Способ может дополнительно включать: извлечение флага переполнения из символа первого типа символа, указывающего, содержится ли часть значения данных символа первого типа символа в последующем третьем типе символа.

Способ может дополнительно включать извлечение флага из каждого символа, указывающего следующий тип символа, который ожидается в наборе символов.

Можно предположить, что начальный символ является символом первого типа, так что начальный символ анализируется для получения по меньшей мере части значения данных.

Этап декодирования закодированного потока битов может включать применение операции кодирования Хаффмана к закодированному потоку битов для генерации набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений остаточных данных, при этом этап применения операции кодирования длин серий для генерации остаточных данных выполняется над наборами символов.

Операция кодирования Хаффмана может быть канонической операцией кодирования Хаффмана.

Способ может дополнительно включать извлечение флага из метаданных конфигурации, сопровождающих закодированный поток битов, указывающего, содержит ли закодированный поток битов байтовый поток, закодированный по длине серии, или данные, закодированные по Хаффману; и выборочное применение операции кодирования Хаффмана на основе флага.

Способ может дополнительно включать: идентификацию символа начального типа, который, как ожидается, будет выведен в кодированном потоке битов; применение операции кодирования Хаффмана к закодированному потоку битов для получения начального символа на основе набора параметров кодирования, связанных с начальным типом ожидаемого символа; извлечение флага из начального символа, указывающего следующий тип символа, который, как ожидается, будет получен из потока битов; и, кроме того, применение операции кодирования Хаффмана к потоку битов для получения последующего символа на основе набора параметров кодирования, связанных с последующим типом символа, который,

как ожидается, будет получен из потока битов.

Способ может дополнительно включать итеративное извлечение флага из декодированного символа, указывающего следующий тип символа, который, как ожидается, будет получен из потока битов, и применение операции кодирования Хаффмана к потоку битов для получения следующего символа на основе набора параметров кодирования, связанных с последующим типом символа, который, как ожидается, будет получен из потока битов.

Таким образом, способ может включать этапы декодирования символа с использованием операции кодирования Хаффмана, декодирования символа с использованием операции кодирования длин серий, идентификации типа символа, ожидаемого следующим из операции кодирования длин серий, и декодирования следующего символа с использованием операции кодирования Хаффмана.

Способ может дополнительно включать: извлечение заголовка потока, содержащего указание множества длин кода, которые должны использоваться для операции канонического кодирования Хаффмана; ассоциирование каждой длины кода с соответствующим символом в соответствии с канонической операцией кодирования Хаффмана; и идентификацию кода, связанного с каждым символом, на основе длин кода в соответствии с канонической операцией кодирования Хаффмана. Таким образом, коды могут быть связаны с последовательными символами, длины кода которых идентичны.

Заголовок потока может содержать указание символа, связанного с соответствующей длиной кода из множества длин кода, и этап связывания каждой длины кода с символом включает связывание каждой длины кода с соответствующим символом в заголовке потока.

Этап связывания каждой длины кода с соответствующим символом включает связывание каждой длины кода с символом из набора заранее определенных символов в соответствии с порядком, в котором была извлечена каждая длина кода.

Способ может дополнительно содержать не связывание символа из набора заранее определенных символов с соответствующей длиной кода, где флаг заголовка потока указывает, что длина кода не существует в заголовке потока для этого символа.

Согласно аспекту изобретения предоставляется способ кодирования видеосигнала. Способ включает: получение входного кадра; обработку входного кадра для генерации остаточных данных, остаточные данные, позволяющие декодеру восстановить входной кадр из опорного восстановленного кадра; и применение операции кодирования Хаффмана к остаточным данным, при этом операция Хаффмана содержит генерирование кодированного битового потока, содержащего набор кодов, кодирующих набор символов, представляющих остаточные данные.

Согласно дополнительному аспекту изобретения может быть предоставлен способ декодирования видеосигнала, при этом способ включает: извлечение закодированного битового потока; декодирование кодированного битового потока для генерирования остаточных данных, и восстановление оригинального кадра видеосигнала из остаточных данных и опорный восстановленный кадр, в котором этап декодирования кодированного битового потока содержит: применение операции кодирования Хаффмана для генерирования остаточных данных, отличающееся тем, что операция кодирования Хаффмана включает генерирование набора символов, представляющих остаточные данные путем сравнения кодированного битового потока с опорным отображением кодов в соответствующем символе, генерирование остаточных данных на основе значений данных, отличающихся от нуля, и подсчет последовательных нулевых значений.

Согласно дополнительному аспекту может быть предоставлено устройство для кодирования набора данных в кодированный набор данных. Устройство выполнено с возможностью кодирования входного видео в соответствии с вышеуказанными этапами. Устройство может содержать процессор, выполненный с возможностью выполнения способа в соответствии с любым из вышеуказанных аспектов.

Согласно дополнительному аспекту может быть предоставлено устройство для декодирования набора данных в восстановленное видео из набора данных. Устройство выполнено с возможностью декодирования выходного видео в соответствии с вышеуказанными этапами. Устройство может содержать процессор, выполненный с возможностью выполнения способа в соответствии с любым из вышеупомянутых аспектов.

Также могут быть предоставлены кодер и декодер.

Согласно дополнительным аспектам изобретения могут быть предоставлены машиночитаемые носители, которые при исполнении процессором заставляют процессор выполнять любой из способов в соответствии с вышеупомянутыми аспектами.

Согласно дополнительному аспекту может быть предоставлен закодированный битовый поток, содержащий закодированную версию остаточных данных для расширенного потока, причем остаточные данные выполнены с возможностью использования с опорным восстановленным кадром для восстановления исходного кадра видеосигнала, при этом закодированные остаточные данные содержат результат применения операции кодирования длин серий к остаточным данным, при этом операция кодирования длин серий приводит к получению набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений остаточных данных; причем набор символов анализируют декодером для получения ненулевых значений

данных остаточных данных и подсчетов последовательных нулевых значений, при этом ненулевые значения данных и подсчеты последовательных нулевых значений выполнены с возможностью использования для получения остаточных данных на декодере, причем закодированный битовый поток характеризуется следующим: метаданные конфигурации содержат флаг, указывающий, представляет ли закодированный битовый поток байтовый поток, закодированный по длине серий, или данные, закодированные по Хаффману, причем в ответ на флаг, указывающий, что закодированный битовый поток содержит данные, закодированные по Хаффману, набор символов, представляющих ненулевые значения данных остаточных данных, и набор символов, представляющих подсчеты последовательных нулевых значений, дополнительно кодируют по Хаффману в закодированном битовом потоке; и причем флаг выполнен с возможностью использования для указания декодеру применить операцию кодирования Хаффмана к закодированному битовому потоку для генерирования набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений до применения операции кодирования длин серий на декодере.

Подробное описание изобретения

Примеры систем и способов в соответствии с изобретением теперь будут описаны со ссылкой на прилагаемые чертежи, на которых:

на фиг. 1 проиллюстрирована высокоуровневая схема процесса кодирования;

на фиг. 2 проиллюстрирована высокоуровневая схема процесса декодирования;

на фиг. 3 проиллюстрирована высокоуровневая схема процесса генерирования остаточных данных;

На фиг. 4 проиллюстрирована высокоуровневая схема процесса генерирования дополнительных остаточных данных с другим уровнем качества;

на фиг. 5 проиллюстрирована высокоуровневая схема процесса восстановления кадра из остаточных данных;

на фиг. 6 проиллюстрирована иерархическая структура данных;

на фиг. 7 проиллюстрирована дополнительная иерархическая структура данных;

на фиг. 8 проиллюстрирован пример процесса кодирования;

на фиг. 9 проиллюстрирован пример процесса декодирования;

на фиг. 10 проиллюстрирована структура первого символа данных;

на фиг. 11 проиллюстрирована структура второго символа данных;

на фиг. 12 проиллюстрирована структура символа запуска;

на фиг. 13 проиллюстрирована структура заголовка первого потока;

на фиг. 14 проиллюстрирована структура заголовка второго потока;

на фиг. 15 проиллюстрирована структура заголовка третьего потока;

на фиг. 16 проиллюстрирована структура заголовка четвертого потока;

на фиг. 17A-17E проиллюстрированы деревья Хаффмана и

на фиг. 18 проиллюстрирован конечный автомат кодирования длин серий.

Настоящее изобретение соотносится со способами. В частности, настоящее изобретение связано со способами кодирования и декодирования сигналов. Способы обработки данных могут включать, помимо прочего, получение, извлечение, вывод, прием и восстановление данных.

Рассматриваемая в этом документе технология кодирования является гибким, адаптируемым, высокоэффективным и недорогим в вычислительном отношении форматом кодирования, который сочетает в себе формат кодирования видео, базовый кодек (например, AVC, HEVC или любой другой существующий или будущий кодек) с уровнем расширения кодированных данных, закодированных с использованием других технологий.

Применяемая технология использует исходный сигнал с пониженной дискретизацией, закодированный с помощью базового кодека, для формирования базового потока. Расширенный поток формируется с использованием кодированного набора остаточных данных, которые исправляют или улучшают базовый поток, например, путем увеличения разрешения или увеличения частоты кадров. В иерархической структуре может быть несколько уровней расширенных данных. Стоит отметить, что обычно ожидается, что базовый поток будет декодируемым аппаратным декодером, в то время как ожидается, что поток расширения будет подходить для реализации программной обработки с подходящим потреблением энергии.

Необходимы способы и системы для эффективной передачи и хранения расширенной закодированной информации.

Важно, чтобы любая операция энтропийного кодирования, используемая в новой технологии кодирования, была адаптирована к конкретным требованиям или ограничениям потока расширения и имела низкую сложность. Такие требования или ограничения включают в себя: потенциальное снижение вычислительных возможностей в результате необходимости программного декодирования потока расширения; необходимость комбинирования декодированного набора остаточных данных с декодированным кадром; вероятная структура остаточных данных, то есть относительно высокая доля нулевых значений с сильно изменчивыми значениями данных в большом диапазоне; нюансы входного квантованного блока коэффициентов; и структура потока расширения, которая представляет собой набор дискретных оста-

точных кадров, разделенных на плоскости, слои и т.п. Энтропийное кодирование также должно подходить для множества уровней расширения в потоке расширения.

Исследование, проведенное изобретателями, установило, что современные схемы энтропийного кодирования, используемые в видео, такие как контекстно-зависимое адаптивное двоичное арифметическое кодирование (CABAC) или контекстно-адаптивное кодирование с переменной длиной слова (CAVLC), вряд ли будут подходящими. Например, механизмы прогнозирования могут быть ненужными или могут не давать достаточных преимуществ, чтобы компенсировать их вычислительную нагрузку, учитывая структуру входных данных. Более того, арифметическое кодирование, как правило, требует больших вычислительных затрат, и реализация соответствующего программного обеспечения зачастую нежелательна.

Обратите внимание, что ограничения, наложенные на поток расширения, означают, что простая и быстрая операция энтропийного кодирования важна для того, чтобы поток расширения мог эффективно корректировать или улучшать отдельные кадры базового декодированного видео. Обратите внимание, что в некоторых сценариях базовый поток также декодируется практически одновременно перед объединением, что создает нагрузку на ресурсы.

Настоящий документ предпочтительно соответствует требованиям следующих документов ISO/IEC: "Call for Proposals for Low Complexity Video Coding Enhancements" ISO/IEC JTC1/SC29/WG11 N17944, Макао, Китай, окт. 2018 и "Requirements for Low Complexity Video Coding Enhancements" ISO/IEC JTC1/SC29/WG11 N18098, Макао, Китай, окт. 2018. Более того, описанные здесь подходы могут быть включены в продукты PERSEUS®, поставляемые компанией V-Nova International Ltd.

Общая структура предлагаемой схемы кодирования, в которой могут быть применены описанные в этом документе способы, использует исходный сигнал с пониженной дискретизацией, закодированный с помощью базового кодека, добавляет первый уровень данных коррекции к декодированному выходному сигналу базового кодека для генерации скорректированного изображения, а затем добавляет дополнительный уровень данных расширения к версии скорректированного изображения с повышенной дискретизацией.

Таким образом, потоки рассматриваются как базовый поток и поток расширения. Стоит отметить, что обычно ожидается, что базовый поток будет декодироваться аппаратным декодером, в то время как ожидается, что поток расширения будет подходить для реализации программной обработкой с подходящим энергопотреблением.

Эта структура создает множество степеней свободы, которые обеспечивают большую гибкость и адаптируемость ко многим ситуациям, что делает формат кодирования подходящим для многих случаев использования, включая передачу Over-The-Top (OTT), прямую трансляцию, прямую трансляцию Ultra High Definition (UHD) и т.п.

Хотя декодированный вывод базового кодека не предназначен для просмотра, это полностью декодированное видео с более низким разрешением, что делает вывод совместимым с существующими декодерами и, где это считается подходящим, также может использоваться в качестве вывода с более низким разрешением.

В общем случае остаточные данные относятся к разнице между значением опорного массива или опорного кадра и фактическим массивом или кадром данных. Следует отметить, что этот обобщенный пример не зависит от выполняемых операций кодирования и характера входного сигнала. Ссылка на "остаточные данные", в контексте данного документа, относится к данным, полученным из набора остаточных данных, например набор самих остаточных данных или результат набора операций обработки данных, которые выполняются над набором остаточных данных.

В некоторых примерах, представленных в данном описании, ряд кодированных потоков может быть сгенерирован и независимо отправлен декодеру. То есть в примерах способов кодирования сигнала сигнал может быть закодирован с использованием по меньшей мере двух уровней кодирования. Первый уровень кодирования может выполняться с использованием первого алгоритма кодирования, а второй уровень может быть кодирован с использованием второго алгоритма кодирования. Способ может включать: получение первой части битового потока путем кодирования первого уровня сигнала; получение второй части битового потока путем кодирования второго уровня сигнала; и передачу первой части битового потока и второй части потока байтов как двух независимых потоков битов.

Следует отметить, что метод энтропийного кодирования, предлагаемый в данном документе, не ограничивается множеством LoQ, проиллюстрированных на фигурах, но обеспечивает полезность любых остаточных данных, используемых для обеспечения расширения или исправления кадра видео, восстановленного из кодированного потока, закодированного с использованием устаревшей технологии кодирования видео, такой как HEVC. Например, полезность метода кодирования при кодировании остаточных данных различных LoQ особенно выгодна.

Отметим, что на фиг. 1-5, проиллюстрированы примеры кодирования и декодирования схем, в которых методы энтропийного кодирования, представленные в данном документе, могут обеспечить полезность, но будет понятно, что метод кодирования может обычно использоваться для кодирования остаточных данных.

Некоторые примеры, представленные в данном описании, относятся к обобщенному процессу кодирования и декодирования, который обеспечивает иерархическую, масштабируемую гибкую технологию кодирования. Первая часть битового потока или первый независимый битовый поток могут быть декодированы с использованием первого алгоритма декодирования, а вторая часть битового потока или второго либо независимого битового потока может быть декодирована с использованием второго алгоритма декодирования. Первый алгоритм декодирования может быть декодирован традиционным декодером с использованием устаревшего оборудования.

Возвращаясь к начальному процессу, описанному выше и обеспечивающему базовый поток и два уровня расширения в потоке расширения, пример процесса обобщенного кодирования показан на блок-схеме, проиллюстрированной на фиг. 1. Входное видео с полным разрешением 100 обрабатывается для генерирования различных кодированных потоков 101, 102, 103. Первый кодированный поток (кодированный базовый поток) создается путем подачи в базовый кодек (например, AVC, HEVC или любой другой кодек) версии входного видео с пониженной дискретизацией. Кодированный базовый поток может называться базовым слоем или базовым уровнем. Второй закодированный поток (закодированный поток уровня 1) создается путем обработки остаточных данных, полученных путем обработки разницы между восстановленным базовым видео кодеком и версией входного видео с пониженной дискретизацией. Третий кодированный поток (закодированный поток уровня 0) создается путем обработки остаточных данных, полученных путем обработки разницы между версией с повышенной дискретизацией скорректированной версии восстановленного базового кодированного видео и входным видео.

Операция уменьшения дискретизации может применяться к входному видео для создания видео с пониженной дискретизацией, которое должно быть закодировано с помощью базового кодека. Уменьшение дискретизации может выполняться как в вертикальном, так и в горизонтальном направлениях или, альтернативно, только в горизонтальном направлении.

Каждый процесс кодирования потока расширения не обязательно может предусматривать этап увеличения дискретизации. На фиг. 1, например, первый поток расширения концептуально является потоком коррекции, тогда как второй поток расширения подвергается увеличению дискретизации, чтобы обеспечить уровень расширения.

Рассматривая процесс генерирования расширенных потоков более подробно, для генерации кодированного потока уровня 1 кодированный базовый поток декодируется 114 (т.е. операция декодирования применяется к кодированному базовому потоку для генерирования декодированного базового потока). Затем создается разница между декодированным базовым потоком и входным видео с пониженной дискретизацией 110 (т.е. операция вычитания применяется к входному видео с пониженной дискретизацией и декодированному базовому потоку для генерирования первого набора остаточных данных).

В данном документе термин "остаточные данные" используется так же, как и термин, известный в данной области техники, то есть обозначает ошибку между опорным кадром и желаемым кадром. В этом документе опорный кадр представляет собой декодированный базовый поток, а желаемый кадр представляет собой входное видео с пониженной дискретизацией. Таким образом, остаточные данные, используемые на первом уровне расширения, можно рассматривать как скорректированное видео, поскольку они "корректируют" декодированный базовый поток во входное видео с пониженной дискретизацией, которое использовалось в операции базового кодирования.

Опять же, описанная ниже операция энтропийного кодирования подходит для любых остаточных данных, например любых данных, связанных с набором остаточных данных.

Затем разность кодируется 115 для генерирования закодированного потока уровня 1 102 (т.е. операция кодирования применяется к первому набору остаточных данных для генерирования первого потока расширения).

Как отмечено выше в документе, расширенный поток может содержать первый уровень расширения 102 и второй уровень расширения 103. Первый уровень расширения 102 может рассматриваться как скорректированный поток. Второй уровень расширения 103 можно рассматривать как дополнительный уровень расширения, который преобразует скорректированный поток в исходное входное видео.

Дополнительный уровень расширения 103 создается путем кодирования дополнительного набора остаточных данных, которые представляют собой разницу 119 между версией 117 с повышенной дискретизацией декодированного потока уровня 1 118 и входным видео 100.

Как отмечалось в этом документе, поток с повышенной дискретизацией сравнивается с входным видео, в результате создается дополнительный набор остаточных данных (т.е. операция разности применяется к повторно созданному потоку с повышенной дискретизацией, чтобы сгенерировать дополнительный набор остаточных данных). Дальнейший набор остаточных данных затем кодируется 121 как закодированный поток расширения уровня 0 (т.е. операция кодирования затем применяется к дополнительному набору остаточных данных для генерации закодированного дополнительного потока расширения).

Таким образом, как проиллюстрировано на фиг. 1 и описано выше в этом документе, выходом процесса кодирования является базовый поток 101 и один или более потоков 102, 103 расширения, которые предпочтительно содержат первый уровень расширения и дополнительный уровень расширения.

Соответствующий обобщенный процесс декодирования изображен на блок-схеме, проиллюстриро-

ванной на фиг. 2. Декодер принимает три потока 101, 102, 103, сгенерированные кодером, вместе с заголовками, содержащими дополнительную информацию декодирования. Кодированный базовый поток декодируется базовым декодером, соответствующим базовому кодеку, используемому в кодере, и его выходные данные объединяются с декодированными остаточными данными, полученными из кодированного потока уровня 1. Комбинированное видео подвергается повышающей дискретизации и дополнительно комбинируется с декодированными остаточными данными, полученными из кодированного потока уровня 0.

В процессе декодирования декодер может анализировать заголовки (глобальная конфигурация, конфигурация изображения, блок данных) и настраивать декодер на основе этих заголовков. Чтобы воссоздать входное видео, декодер может декодировать каждый из базового потока, первого потока расширения и дополнительного потока расширения. Кадры потока могут быть синхронизированы, а затем объединены для получения декодированного видео.

Проиллюстрированные на каждой из фиг. 1 и 2 операции кодирования уровня 0 и уровня 1 могут включать в себя этапы преобразования, квантования и энтропийного кодирования. Точно так же на этапе декодирования остаточные данные могут быть пропущены через энтропийный декодер, деквантизатор и модуль обратного преобразования. Может использоваться любое подходящее кодирование и соответствующая операция декодирования. Однако предпочтительно, чтобы этапы кодирования уровня 0 и уровня 1 могли выполняться программно.

Таким образом, приведенные здесь способы и устройства основаны на общем алгоритме, который построен на существующем алгоритме кодирования и/или декодирования (таком как стандарты MPEG, такие как AVC/H.264, HEVC/H.265 и т.п., а также нестандартный алгоритм, такой как VP9, AV1 и другие), который работает как базовый уровень для слоя расширения, который работает в соответствии с другим алгоритмом кодирования и/или декодирования. Идея, лежащая в основе общего алгоритма, заключается в иерархическом кодировании/декодировании видеокадра в отличие от блочных подходов, используемых в семействе алгоритмов MPEG. Иерархическое кодирование кадра включает в себя создание остаточных данных для полного кадра, затем прореженного кадра и так далее.

Остаточные данные сжатия видео для полноразмерного видеокадра могут называться LoQ-0 (например, 1920×1080 для видеокадра HD), тогда как данные прореженного кадра могут называться LoQ-x, где x обозначает количество иерархических прореживаний. В примерах, проиллюстрированных на фиг. 1 и 2, переменная x имеет максимальное значение 1 и, следовательно, существует 2 иерархических уровня, для которых будут сгенерированы остаточные данные сжатия.

На фиг. 3 проиллюстрирован пример того, как LoQ-1 может быть сгенерирован на устройстве кодирования. На данной фигуре общий алгоритм и способы описаны с использованием алгоритма кодирования/декодирования AVC/H.264 в качестве базового алгоритма, но понятно, что другие алгоритмы кодирования/декодирования могут использоваться в качестве базовых алгоритмов без какого-либо влияния на способ, посредством которого работает общий алгоритм.

Конечно, будет понятно, что блоки, проиллюстрированные на фиг. 3, являются просто примерами того, как можно реализовать широкие концепции.

Схема, проиллюстрированная на фиг. 3, демонстрирует процесс генерирования энтропийного кодирования остаточных данных для уровня иерархии LoQ-1. В этом примере первым шагом является прореживание входящего несжатого видео с коэффициентом 2. Этот прореженный кадр затем пропускается через базовый алгоритм кодирования (в данном случае алгоритм кодирования AVC/H.264), при этом создается и сохраняется ссылка с энтропийным кодированием на кадр. Затем генерируется декодированная версия закодированного опорного кадра, а разница между декодированным опорным кадром и прореженным кадром (остаточные данные LoQ-1) будет формировать входные данные для преобразования блока.

Преобразование (например, преобразование на основе Адамара в этом проиллюстрированном примере) преобразует эту разницу в 4 компонента (или плоскости), а именно A (средний), H (горизонтальный), V (вертикальный) и D (диагональный). Затем эти компоненты квантуются с использованием переменных, называемых "шириной шага" (например, остаточное значение может быть разделено на ширину шага и выбранное ближайшее целое значение). Подходящий процесс энтропийного кодирования является предметом этого изобретения и подробно описывается ниже в этом документе. Эти квантованные остаточные данные затем энтропийно кодируются, чтобы удалить любую избыточную информацию. Квантованные кодированные коэффициенты или компоненты (Ae, He, Ve и De) затем помещаются в последовательный поток с пакетами определения, вставленными в начало потока, этот заключительный этап выполняется с использованием процедуры сериализации файлов. Пакетные данные могут включать в себя такую информацию, как спецификация кодера, тип применяемой повышающей дискретизации, отбрасываются ли плоскости A и D или нет, и другую информацию, позволяющую декодеру декодировать потоки.

Как опорные данные (кадр с базовым энтропийным кодированием половинного размера), так и остаточные данные LoQ-1 с энтропийным кодированием могут быть буферизованы, переданы или сохранены для использования декодером во время процесса восстановления.

Чтобы сгенерировать остаточные данные LoQ-0, квантованный вывод разветвляется, и над ним выполняются процессы обратного квантования и преобразования, чтобы восстановить остаточные данные LoQ-1, которые затем добавляются к декодированным опорным данным (кодируются и декодируются) чтобы получить обратно видеокадр, очень похожий на первоначально прореженный входной кадр.

Этот процесс идеально имитирует процесс декодирования, и, следовательно, первоначально прореженный кадр не используется.

На фиг. 4 проиллюстрирован пример того, как LoQ-0 может быть сгенерирован на устройстве кодирования. Чтобы получить остаточные данные LoQ-0, восстанавливается кадр размера LoQ-1, как описано в предыдущем разделе.

Следующим шагом является выполнение повышающей дискретизации восстановленного кадра до полного размера (на 2). На этом этапе могут использоваться различные алгоритмы для улучшения процесса повышения дискретизации, такие как алгоритмы ближайшего, билинейного, точного или кубического алгоритмов. Этот восстановленный полноразмерный кадр, называемый "прогнозируемым" кадром, затем вычитается из исходного несжатого видеовхода, что создает некоторые остаточные данные (остаточные данные LoQ-1).

Подобно процессу LoQ-1, разница затем преобразуется, квантуется, энтропийно кодируется и файл сериализуется, что затем приводит к формированию третьих и последних данных. Это может быть буферизовано, передано или сохранено для дальнейшего использования декодером. Как можно видеть, компонент, называемый "прогнозируемое среднее" (описанный ниже), может быть получен с использованием процесса повышения дискретизации и использован вместо компонента A (среднего) для дальнейшего повышения эффективности алгоритма кодирования.

На фиг. 5 схематически проиллюстрировано, как процесс декодирования может быть выполнен в конкретном примере. Энтропийно закодированные данные, энтропийно закодированные остаточные данные LOQ-1 и энтропийно закодированные остаточные данные LOQ-O (например, в виде сериализованных в файл данных). Данные энтропийного кодирования содержат базу кодирования уменьшенного размера (например, половинного размера, т.е. с размерами W/2 и H/2 по отношению к полному кадру, имеющему размеры W и H).

Затем данные, закодированные энтропийным кодированием, декодируются с использованием алгоритма декодирования, соответствующего алгоритму, который использовался для кодирования этих данных (в примере, алгоритм декодирования AVC/H.264). В конце этого шага создается декодированный видеокадр уменьшенного размера (например, половинного размера) (обозначенный в настоящем примере как видео AVC/H.264).

Параллельно декодируются остаточные данные с энтропийным кодированием LoQ-1. Как рассматривалось ранее в этом документе, остаточные данные LoQ-1 кодируются с использованием четырех коэффициентов или компонентов (A, V, H и D), которые, как проиллюстрировано на этой фигуре, имеют размеры, соответствующие одной четверти размера полного кадра, а именно W/4 и H/4. Это связано с тем, что, как рассматривалось в предыдущих патентных заявках US 13/893669 и PCT/EP2013/059847, содержание которых включено в настоящий документ посредством ссылки, четыре компонента содержат всю информацию, относящуюся к остаточным данным, и генерируются путем применения преобразования 2×2 ядра к остаточным данным, размерность которых для LoQ-1 будет W/2 и H/2, то есть такая же размерность данных, закодированных энтропией уменьшенного размера. Четыре компонента энтропийно декодируются, затем деквантизируются и, наконец, преобразуются обратно в исходные остаточные данные LoQ-1 с помощью обратного преобразования (в данном случае обратного преобразования Адамара 2×2).

Декодированные остаточные данные LoQ-1 затем добавляются к декодированному видеокадру для создания восстановленного видеокадра уменьшенного размера (в данном случае половинного размера), идентифицированного как восстановление размера Half-2D.

Этот восстановленный видеокадр затем подвергается повышающей дискретизации, которая доводит его до полного разрешения (так, в этом примере, от полуширины (W/2) и полувысоты (H/2) до полной ширины (W) и полной высоты (H)) с использованием фильтра с повышающей дискретизацией, такого как билинейный, бикубический, точный и т.п. Восстановленный видеокадр с повышенной дискретизацией будет предсказанным кадром (полноразмерным, WxH), к которому затем добавляются остаточные данные декодирования LoQ-0.

В частности, декодируются остаточные данные, закодированные в LoQ-0. Как рассматривалось ранее в этом документе, остаточные данные LoQ-0 кодируются с использованием четырех коэффициентов или компонентов (A, V, H и D), размер которых, как проиллюстрировано на этой фигуре, составляет половину размера полного кадра, а именно W/2 и H/2. Это связано с тем, что, как рассматривалось в предыдущих патентных заявках US 13/893669 и PCT/EP2013/059847, содержание которых включено в настоящий документ посредством ссылки, четыре компонента содержат всю информацию, относящуюся к остаточным данным, и генерируются путем применения преобразования 2×2 ядра к остаточным данным, размерность которых для LoQ-0 была бы W и H, то есть такая же размерность полного кадра. Четыре

компонента энтропийно декодируются (см. процесс, описанный далее в документе), затем деквантизируются и, наконец, преобразуются обратно в исходные остаточные данные LoQ-0 с помощью обратного преобразования (в данном случае обратного преобразования Адамара 2×2).

Декодированные остаточные данные LoQ-0 затем добавляются к предсказанному кадру для создания восстановленного полного видеокадра - выходного кадра.

Структура данных проиллюстрирована в качестве примера на фиг. 6. Как рассматривалось ранее в этом документе, приведенное выше описание было сделано со ссылкой на конкретные размеры и базовые алгоритмы, но вышеупомянутые способы применимы к другим размерам и/или базовым алгоритмам, и приведенное выше описание дается лишь в качестве примера описанных более общих понятий.

В алгоритме кодирования/декодирования, описанном выше в этом документе, обычно существуют 3 плоскости (например, YUV или RGB) с двумя уровнями качества LoQ, которые описываются как LoQ-0 (или верхний уровень, полное разрешение) и LoQ-1 (или нижний уровень, разрешение уменьшенного размера, например, половинное разрешение) в каждой плоскости. Каждая плоскость может представлять различную цветовую составляющую для видеосигнала.

Каждый LoQ содержит четыре компонента или слоя, а именно A, H, V и D. Это дает всего 3×2×4=24 поверхности, из которых 12 полноразмерных (например, W×H) и 12 уменьшенных размеров (например, W/2×H/2). Каждый слой может содержать значения коэффициентов для конкретного одного из компонентов, например слой A может содержать значение коэффициента A в верхнем левом углу для каждого блока 2×2 входного изображения или кадра.

На фиг. 7 проиллюстрирован альтернативный вид предлагаемой иерархической структуры данных. Закодированные данные могут быть разделены на части. Каждую полезную нагрузку можно иерархически упорядочить по кускам. То есть каждая полезная нагрузка группируется в плоскости, затем в каждой плоскости каждого уровня группируется в слои, и каждый слой содержит набор фрагментов для этого слоя. Уровень представляет каждый уровень расширения (первый или последующий), а слой представляет собой набор коэффициентов преобразования.

Способ может включать извлечение фрагментов для двух уровней расширения для каждой плоскости. Способ может содержать извлечение 16 слоев для каждого уровня (например, если используется преобразование 4×4). Таким образом, каждая полезная нагрузка упорядочивается в набор фрагментов для всех слоев на каждом уровне, а затем набор фрагментов для всех слоев на следующем уровне плоскости. Затем полезная нагрузка включает набор фрагментов для слоев первого уровня следующей плоскости и так далее.

Таким образом, способ может декодировать заголовки и выводить энтропийно кодированные коэффициенты, сгруппированные по плоскости, уровням и слоям, принадлежащим декодируемому расширению изображения. Таким образом, на выходе могут быть поверхности массива (n Плоскостей)×(n Уровень)×(n Слой) с элементами поверхности [n Плоскостей][n Уровень][n Слой].

Отметим, что способы энтропийного кодирования, представленные в данном документе, могут выполняться для каждой группы, то есть могут выполняться для каждой поверхности, для каждой плоскости, для каждого уровня (LoQ) или для каждого отдельного слоя. Отметим, что энтропийное кодирование может выполняться для любых остаточных данных и не обязательно квантованных и/или преобразованных коэффициентов.

Как проиллюстрировано на фиг. 4 и 5, предлагаемые операции кодирования и декодирования предусматривают этап или операцию энтропийного кодирования. В дальнейшем предлагается, чтобы операция энтропийного кодирования содержала один или более из компонента кодирования длин серий (RLE) и компонента кодирования Хаффмана. Эти два компонента могут взаимодействовать друг с другом, обеспечивая дополнительные преимущества.

Как отмечено и проиллюстрировано на фиг. 8, вход энтропийного кодера представляет собой поверхность (например, остаточные данные, полученные на основе квантованного набора остаточных данных, как проиллюстрировано в этом примере), а выходом процесса является энтропийно-кодированная версия остаточных данных (Ae, He, Ve, De). Однако, как указано выше в этом документе, следует отметить, что энтропийное кодирование может выполняться для любых остаточных данных и не обязательно квантованных и/или преобразованных коэффициентов. На фиг. 9 проиллюстрирован соответствующий высокоуровневый декодер с инверсными входами и выходами. То есть энтропийный декодер принимает в качестве входных данных энтропийно-кодированные остаточные данные (Ae, He, Ve, De) и выводит остаточные данные (например, квантованные остаточные данные в этом проиллюстрированном примере).

Обратите внимание, что как проиллюстрировано на фиг. 8, обычно показано декодирование RLE до декодирования по Хаффману, но, как будет ясно из всей настоящей заявки, такой порядок не является ограничивающим, и этапы могут быть взаимозаменяемыми, взаимосвязанными или располагаться в любом порядке.

Точно так же мы отмечаем, что кодирование длин серий может быть предоставлено без кодирования Хаффмана, и в сравнительных случаях кодирование длин серий и кодирование Хаффмана может не выполняться (либо заменено альтернативным энтропийным кодированием без потерь, либо энтропийное

кодирование не выполняется вообще). Например, в продуктивном примере с уменьшенным акцентом на сжатие данных конвейер кодирования может не содержать энтропийное кодирование, но преимущества остаточных данных могут заключаться в многоуровневом хранилище для распространения.

Кодер сравнительных длин серий (RLE) может сжимать данные путем кодирования последовательностей того же значения данных, что и одно значение данных, и счет этого значения данных. Например, последовательность 555500022 может быть закодирована как (4,5) (3,0) (2,2). То есть идет серия из четырех пятерок, серия из трех нулей, за которыми следует серия из двух двоек.

Для кодирования остаточных данных предлагается модифицированная операция кодирования RLE. Чтобы воспользоваться структурой остаточных данных, предлагается кодировать только серии нулей. То есть каждое значение отправляется как величина, при этом каждый ноль отправляется как серия нулей. Операции модифицированного кодирования RLE, описанные в данном документе, могут использоваться, например для обеспечения энтропийного кодирования и/или декодирования на одном или более слоях LoQ, например как показано на фиг. 1-5.

Таким образом, операция энтропийного кодирования включает анализ остаточных данных и кодирование нулевых значений как серию последовательных нулей.

Как отмечено выше в данном документе, чтобы обеспечить дополнительный уровень сжатия данных (который может быть без потерь), операция энтропийного кодирования дополнительно применяет операцию кодирования Хаффмана к данным с серийным кодированием.

Кодирование Хаффмана и кодирование длин серий ранее сочетались вместе, например, в факсимильном кодировании (например, согласно Рекомендациям ITU T.4 и T.45) и формате обмена файлами JPEG, но за последние 30 лет они были заменены другими способами кодирования. В настоящем описании предлагаются способы и технологии для реализации кодирования Хаффмана в сочетании с кодированием RLE, методы и способы для эффективного обмена данными и метаданными между кодером и декодером, а также методы и способы для уменьшения общего размера данных такой комбинации при использовании для кодирования остаточных данных.

Код Хаффмана представляет собой оптимальный префиксный код, который используется для сжатия данных (например, сжатие без потерь). Префиксный код представляет собой систему кодов, для которой в системе нет работы кода, которая является префиксом любого другого кода. То есть операция кодирования Хаффмана берет набор входных символов и преобразует каждый входной символ в соответствующий код. Выбранный код основан на частоте появления каждого символа в исходном наборе данных. Таким образом, меньшие коды могут быть связаны с символами, которые встречаются чаще, чтобы уменьшить общий размер данных.

Чтобы облегчить кодирование Хаффмана, входные данные предпочтительно структурированы как символы. Выходные данные операции кодирования длин серий предпочтительно структурированы так, чтобы представлять собой поток байтов закодированных данных.

В одном примере операция кодирования длин серий выводит два типа байтов или символов. Первый тип символа представляет собой значение ненулевого пикселя, а второй тип символа представляет собой серию последовательных нулевых значений. То есть количество нулевых значений, которые последовательно встречаются в исходном наборе данных.

В дополнительном примере для кодирования определенных значений данных могут быть объединены два байта или символа. То есть, например, когда значение пикселя больше порогового значения, два байта или символа могут использоваться для кодирования значения данных в символе. Эти два символа могут быть объединены в декодере для воссоздания исходного значения данных пикселя.

Каждый символ или байт может содержать 6 или 7 бит данных и один или более флагов или битов, указывающих тип символа.

В примере осуществления данные, закодированные по длине серий, могут быть эффективно закодированы путем вставки в каждый символ одного или более флагов, указывающих следующий символ, который ожидается в потоке байтов.

Чтобы облегчить синхронизацию между кодером и декодером, первый байт потока данных с кодированием длин серий может быть типом значения данных символа. Затем этот байт может указывать на следующий тип байта, который был закодирован.

В некоторых вариантах воплощения флаг, указывающий следующий байт, может отличаться в зависимости от типа символа. Например, флаг или бит могут быть расположены в начале или в конце байта или в обоих. Например, в начале байта бит может указывать, что следующий символ может быть серией нулей. В конце байта бит может быть битом переполнения, указывающим, что следующий символ содержит часть значения данных, которая должна быть объединена со значением данных в текущем символе.

Далее описывается конкретный пример реализации операции кодирования длин серий. RLE содержит три контекста: RLC_RESIDUAL_LSB, RLC_RESIDUAL_MSB и RLC_ZERO_RUN. Структура этих байтов проиллюстрирована на фиг. 10, 11 и 12.

В первом типе символа, RLC_RESIDUAL_LSB, проиллюстрированном на фиг. 10, кодируются 6 младших значащих битов ненулевого пикселя. Может быть предусмотрен бит выполнения, указываю-

щий, что следующий байт кодирует счет серии нулей. Бит переполнения может быть установлен, если значение пикселя не соответствует 6 битам данных. Когда бит переполнения установлен, контекст следующего байта будет иметь тип RLC_RESIDUAL_MSB. То есть следующий символ будет содержать байт данных, который может быть объединен с битами текущего символа для кодирования значения данных. Если установлен бит переполнения, следующий контекст не может быть серией нулей, и поэтому символ может использоваться для кодирования данных.

На фиг. 10 проиллюстрирован пример этого типа символа. Если значение данных, которые должны быть закодированы, больше порогового значения 64 или значение пикселя не соответствует 6 битам данных, бит переполнения может быть установлен процессом кодирования. Если бит переполнения установлен как младший бит байта, то оставшиеся биты байта могут кодировать данные. Если значение пикселя умещается в пределах 6 битов, бит переполнения не может быть установлен и может быть включен бит выполнения, который указывает, является ли следующий символ значением данных или серией нулей, или нет.

Второй тип символа, RLC_RESIDUAL_MSB, проиллюстрированный на фиг. 11, кодирует биты с 7 по 13 значений пикселей, которые не помещаются в 6 бит данных. Бит 7 этого типа символа кодирует, является ли следующий байт серией нулей или нет.

Третий тип символа, RLC_ZERO_RUN, проиллюстрированный на фиг. 12, кодирует 7 бит нулевого подсчета серий. То есть символ содержит количество последовательных нулей в остаточных данных. Бит выполнения символа имеет высокий уровень, если для кодирования подсчета требуется больше битов. Бит выполнения может быть самым старшим битом байта. То есть, когда имеется ряд последовательных нулей, для которых требуется более 7 доступных битов, например 178 или 255, бит выполнения указывает, что следующий бит будет указывать на дополнительную серию нулей.

Необязательно, дополнительный символ может содержать вторую серию нулей, которая может быть объединена с первой серией, или может содержать набор битов значения, которые могут быть объединены с битами первого символа в декодере для указания счета.

Как указано выше в данном документе, для того, чтобы декодер начал работу в известном контексте, первый символ в кодированном потоке может быть типом значения остаточных данных символа, то есть может быть типом RLC_RESIDUAL_LSB.

В конкретном примере данные RLE могут быть организованы в блоки. Каждый блок может иметь выходную емкость 4096 байт. RLE может переключиться на новый блок в следующих случаях:

текущий блок заполнен;

текущие данные RLE являются прогоном, а в текущем блоке осталось менее 5 байтов; и/или

текущие данные RLE приводят к паре LSB/MSB, а в текущем блоке осталось менее 2 байтов.

Таким образом, операция кодирования длин серий может выполняться над остаточными данными новой технологии кодирования, которая включает кодирование в поток набора значений данных и подсчета последовательных нулевых значений. В конкретной реализации вывод операции кодирования длин серий может быть потоком байтов или символов, где каждый байт является одним из трех типов или контекстов. Байт указывает следующий тип байта, который ожидается в потоке байтов.

Следует отметить, что эти структуры предоставлены в качестве примера, и что могут применяться различные схемы кодирования битов, следуя функциональным принципам, описанным в данном документе. Например, можно обмениваться кодированием младшего и старшего значащих битов и/или можно применять различные длины битов. Кроме того, биты флагов могут быть расположены в различных заранее определенных позициях в байте.

Как отмечено выше в данном документе, операция кодирования Хаффмана может применяться к потоку байтов, чтобы дополнительно уменьшить размер данных. Процесс может создать частотную таблицу относительного появления каждого символа в потоке. Из частотной таблицы процесс может сгенерировать дерево Хаффмана. Код Хаффмана для каждого символа может быть сгенерирован путем обхода дерева от корня до тех пор, пока символ не будет достигнут, назначая бит кода для каждой взятой ветви.

В предпочтительном варианте осуществления для использования со спецификой кодированных по длине серий символов квантованных остатков данных (то есть остаточных данных) может применяться каноническая операция кодирования Хаффмана. Канонические коды Хаффмана могут уменьшить требования к хранению набора кодов в зависимости от структуры входных данных. Каноническая процедура Хаффмана обеспечивает способ создания кода, который неявно содержит информацию о том, какое кодовое слово применимо к какому символу. Коды Хаффмана могут быть установлены с набором условий, например, все коды заданной длины могут иметь лексикографически последовательные значения в том же порядке, что и символы, которые они представляют, а более короткие коды лексикографически предшествуют более длинным кодам. В канонической реализации кодирования Хаффмана только кодовые слова и длины каждого кода должны быть переданы, чтобы декодер мог воспроизвести каждый символ.

После применения операции канонического кодирования Хаффмана размер выходных данных можно сравнить с размером данных после выполнения операции кодирования длин серий. Если размер данных меньше, то меньшие данные могут быть переданы или сохранены. Сравнение размера данных может выполняться на основе размера блока данных, размера слоя, плоскости или поверхности или об-

щего размера кадра или видео. Чтобы сигнализировать декодеру, как данные были закодированы, в заголовке данных может быть передан флаг, что данные кодируются с использованием кодирования Хаффмана, кодирования RLE или того и другого. В альтернативной реализации декодер может иметь возможность определять по характеристикам данных, что данные были закодированы с использованием конкретной операции кодирования. Например, как указано ниже, данные могут быть разделены на заголовок и часть данных, где используется каноническое кодирование Хаффмана, чтобы сигнализировать о длинах кодов закодированных символов.

Канонические данные, закодированные по способу Хаффмана, содержат часть, которая сообщает декодеру о длине кода, используемого для каждого символа, и часть, которая содержит битовый поток, представляющих кодированные данные. В конкретной реализации, предложенной в данном документе, длины кода могут сообщаться в части заголовка, а данные сообщаться в части данных. Предпочтительно часть заголовка будет сигнализироваться для каждой поверхности, но может сигнализироваться для каждого блока или другого подразделения в зависимости от конфигурации.

В предлагаемом примере заголовок может отличаться в зависимости от количества кодируемых ненулевых кодов. Количество кодируемых ненулевых кодов (например, для каждой поверхности или блока) может сравниваться с пороговым значением и заголовком, используемым на основе сравнения. Например, если имеется более 31 символа ненулевого кода, заголовок может последовательно указывать все символы, начиная с заранее определенного сигнала, такого как 0. По порядку, длина каждого символа может сигнализироваться. Если имеется менее 31 ненулевого символа кода, каждое значение символа может сигнализироваться в заголовке с соответствующей длиной кода для этого символа.

На фиг. 13-16 иллюстрируется конкретная реализация форматов заголовков и то, как длины кода могут быть записаны в заголовок потока в зависимости от количества ненулевых кодов.

На фиг. 13 проиллюстрирована ситуация, когда в данных может быть более 31 ненулевого символа. Заголовок включает минимальную длину кода и максимальную длину кода. Длина кода для каждого символа отправляется последовательно. Флаг указывает, что длина символа не равна нулю. Затем биты длины кода отправляются как разница между длиной кода и минимальной сигнальной длиной. Благодаря этому уменьшается общий размер заголовка.

На фиг. 14 проиллюстрирован заголовок, аналогичный заголовку, проиллюстрированному на фиг. 13, но используемый там, где имеется менее 31 ненулевого кода. Заголовок дополнительно включает количество символов в данных, за которым следует значение символа и длина кодового слова для этого символа, снова отправленные как разность.

На фиг. 15 и 16 проиллюстрированы дополнительные заголовки, которые нужно отправлять в особых случаях. Например, если все частоты равны нулю, заголовок потока может быть таким, как проиллюстрировано на фиг. 14, указывая два значения 31 в полях минимальной и максимальной длины, чтобы указать особую ситуацию. Если в дереве Хаффмана есть только один код, заголовок потока может быть таким, как показано на рисунке 16, с использованием значения 0 в полях минимальной и максимальной длины, чтобы указать особую ситуацию, а затем следует значение символа, которое будет использоваться.

В этом последнем примере, где есть только одно значение символа, это может указывать на то, что в остаточных данных есть только одно значение данных.

Таким образом, процесс кодирования можно резюмировать следующим образом:

Анализ остаточных данных для определения значений данных и последовательных подсчетов нулевых значений.

Генерирование набора символов, в котором каждый символ является байтом, а каждый байт содержит либо значение данных, либо серию нулей, а также указание следующего символа в наборе символов. Символы могут содержать бит переполнения, указывающий, что следующий символ включает в себя часть значения данных, включенных в текущий символ, или бит выполнения, указывающий, что следующий символ представляет собой серию нулей.

Преобразование каждого символа фиксированной длины в код переменной, используя каноническое кодирование Хаффмана. Каноническое кодирование Хаффмана анализирует символы, чтобы идентифицировать частоту, с которой каждый символ встречается в наборе символов, и присваивает символу код на основе частоты и значения символа (для одинаковой длины кода).

Генерирование и вывод набора длин кода, каждая из которых связана с символом. Длина кода представляет собой длину каждого кода, назначенного каждому символу.

Объединение кодов переменной длины в битовый поток.

Вывод закодированного битового потока.

В конкретной описанной реализации следует отметить, что максимальная длина кода зависит от количества закодированных символов и количества выборок, используемых для получения таблицы частот Хаффмана. Для N символов максимальная длина кода составляет N-1. Однако, чтобы символ имел k-битный код Хаффмана, количество выборок, используемых для вывода частотной таблицы, должно быть по меньшей мере:

$$S_k = \sum_{i=0}^{k+1} F_i$$

где F_i является i-ым числом Фибоначчи.

Например, для 8-битных символов теоретическая максимальная длина кода составляет 255. Однако для видео HD 1080p максимальное количество выборок, используемых для вывода таблицы частот для контекста RLE остаточных данных, составляет 1920×1080/4=518 400, что находится между S26 и S27. Следовательно, в этом примере ни один символ не может иметь код Хаффмана более 26 бит. Для 4K это число увеличивается до 29 бит.

Для полноты картины на фиг. 17 проиллюстрирован пример кодирования набора символов Хаффманом, в этом примере символы являются буквенно-цифровыми. Как отмечалось выше, код Хаффмана представляет собой оптимальный префиксный код, который может использоваться для сжатия данных без потерь. Префиксный код представляет собой кодовую систему, для которой в системе нет кодового слова, которое является префиксом любого другого кода.

Чтобы найти код Хаффмана для данного набора символов, необходимо создать дерево Хаффмана. Сначала символы сортируются по частоте, например:

Символ	Частота
A	3
B	8
C	10
D	15
E	20
F	43

Два самых нижних элемента удаляются из списка и превращаются в листья с родительским узлом, частота которого равна сумме частот двух нижних элементов. Частичное дерево проиллюстрировано на фиг. 17a.

Новый отсортированный список частот:

Символ	Частота
C	10
*	11
D	15
E	20
F	43

Затем цикл повторяется, объединяя два самых нижних элемента, как проиллюстрировано на фиг. 17b.

Новый список:

Символ	Частота
D	15
E	20
*	21
F	43

Это повторяется до тех пор, пока в списке не останется только один элемент, как проиллюстрировано на фиг. 17c, 17d, 17e и в следующих соответствующих таблицах.

Символ	Частота
*	21
*	35
F	43

Символ	Частота
F	43
*	56

Символ	Частота
*	99

После построения дерева для генерирования кода Хаффмана для символа тройка обходит от корня до этого символа, выводя 0 каждый раз, когда берется левая ветвь, и 1 каждый раз, когда выбирается правая ветвь. В приведенном выше примере это дает следующий код:

Символ	Код	Длина кода
A	1010	3
B	1011	3
C	100	2
D	110	2
E	111	2
F	0	0

Длина кода символа представляет собой длину соответствующего ему кода.

Чтобы декодировать код Хаффмана, дерево просматривается, начиная с корня, выбирая левый путь, если читается 0, и правый путь, если читается 1. Символ обнаруживается при ударе по листу.

Как описано выше в документе, символы RLE могут кодироваться с использованием кодирования Хаффмана. В предпочтительном примере используется каноническое кодирование Хаффмана. В реализации канонического кодирования Хаффмана может использоваться процедура кодирования Хаффмана, и полученный код преобразовывается в канонические коды Хаффмана. То есть таблица перестраивается так, чтобы одинаковые длины были последовательными в том же порядке, что и символы, которые они представляют, и чтобы более поздний код всегда был выше по значению, чем более ранний. Если символы расположены в алфавитном порядке, их можно переставить в алфавитном порядке. Если символы являются значениями, они могут располагаться в последовательном порядке значений, и соответствующие кодовые слова изменяются, чтобы соответствовать указанным выше ограничениям. Каноническое кодирование Хаффмана и/или другие подходы к кодированию Хаффмана, описанные в данном документе, могут отличаться от базового примера, проиллюстрированного на фиг. 17a-17e.

В примерах, описанных выше в данном документе, операция кодирования Хаффмана может применяться к символам данных, кодированным с использованием операции кодирования модифицированной длины серий. Далее предлагается создать частотную таблицу для каждого контекста или состояния RLE. Для каждого типа символа, закодированного с помощью операции кодирования RLE, может быть свой набор кодов Хаффмана. В варианте реализации может быть свой кодировщик Хаффмана для каждого контекста или состояния RLE.

Предполагается, что один заголовок потока может быть отправлен для каждого контекста RLE или типа символа, то есть множество заголовков потока может указывать длины кодов для каждого набора кодов, то есть для каждого кодера Хаффмана или каждой таблицы частот.

Ниже описан пример конкретных этапов реализации кодировщика. В одной реализации кодирование данных, выводимых RLE, выполняется в три основных этапа:

1. Инициализация кодеров (например, с помощью функции InitialiseEncode):

а. инициализация каждого кодера (по одному на состояние RLE) соответствующей таблицей частот, созданной RLE;

б. создание дерева Хаффмана;

в. вычисление длины кода каждого символа; и

г. определение минимальной и максимальной длины кода.

2. Запись длины кода и таблицы кодов в заголовок потока (например, с помощью функции WriteCodeLengths) для каждого кодера:

а. присваивание кодов символам (например, с помощью функции AssignCodes). Обратите внимание, что функция, используемая для присвоения кодов символу, немного отличается от примера, проиллюстрированного на фиг. 19a-19e выше, поскольку она обеспечивает последовательное выполнение всех кодов заданной длины, т.е. использует каноническое кодирование Хаффмана;

б. запись минимальной и максимальной длины кода в заголовок потока; и

в. запись длин кода каждого символа в поток.

3. Кодирование данных RLE:

а. присваивание текущего RLE-контекста RLC_RESIDUAL_LSB;

б. кодирование текущего символа во входном потоке с помощью кодера, соответствующего текущему контексту;

в. использование конечного автомата RLE, чтобы получить следующий контекст (см. фиг. 20). Если это не последний символ потока, перейдите к b; и

г. если данные RLE представляют собой только один блок, а закодированный поток больше, чем RLE, сохранение закодированного потока RLE, а не закодированного потока Хаффмана.

Для каждого блока RLE кодеры Хаффмана создают соответствующую кодированную часть Хаффмана.

Таким образом, результатом описанного здесь процесса кодирования является либо битовый поток, закодированный по Хаффману, либо поток байтов, закодированный по длине серий. Таким образом соответствующий декодер использует соответствующую операцию декодирования Хаффмана способом, обратным операции кодирования. Тем не менее, в описанной ниже операции декодирования есть нюансы для повышения эффективности и адаптации к специфике операции кодирования, описанной выше в документе.

Когда кодер сигнализирует в заголовке или других метаданных конфигурации, что была использована операция кодирования RLE или Хаффмана, декодер может сначала идентифицировать, что операция декодирования Хаффмана должна быть применена к потоку. Точно также, как описано выше, декодер может идентифицировать из потока данных, использовалось ли кодирование Хаффмана, путем определения количества частей потока. Если есть часть заголовка и часть данных, тогда должна применяться операция декодирования Хаффмана. Если в потоке есть только часть данных, тогда может применяться только часть декодирования RLE.

Энтропийный декодер выполняет обратные преобразования декодирования. В данном примере за декодированием по Хаффману следует декодирование длины серии.

В следующем примере описывается процесс декодирования, в котором битовый поток был закодирован с использованием канонической операции кодирования Хаффмана и операции кодирования длин серий, как описано выше в примере потока кодирования. Однако следует понимать, что операции кодирования Хаффмана и операции кодирования длин серий могут применяться отдельно для декодирования потоков, закодированных другим способом.

Вход в процесс представляет собой закодированный битовый поток. Битовый поток может быть получен из сохраненного файла или передан в потоковом режиме локально или удаленно. Закодированный битовый поток представляет собой последовательность последовательных битов без сразу различимой структуры. Только после применения операции кодирования Хаффмана к потоку битов можно вывести последовательность символов.

Таким образом, процесс сначала применяет операцию кодирования Хаффмана к потоку битов.

Предполагается, что первый кодированный символ является символом типа значения данных. Таким образом, если операция кодирования Хаффмана использует разные частотные таблицы или параметры, операция кодирования Хаффмана может использовать правильную кодовую книгу для типа символа.

Каноническая операция кодирования Хаффмана должна сначала извлечь подходящие параметры кодирования или метаданные, чтобы облегчить декодирование битового потока. В приведенном здесь примере параметры представляют собой набор длин кода, извлеченных из заголовка потока. Для простоты будет описан один набор длин кода, которым обмениваются кодер и декодер, но отметим, что, как и выше, могут передаваться сигналы нескольких наборов длин кода, так что операция кодирования может использовать разные параметры для каждого типа символа, который она предполагает расшифровать.

Операция кодирования назначит длины кода, полученные в заголовке потока, соответствующему символу. Как отмечено выше, заголовок потока может отправляться в разных типах. Если заголовок потока включает в себя набор длин кода с соответствующими символами, каждая длина кода может быть связана с соответствующим значением символа. Если длины кода отправляются последовательно, декодер может ассоциировать каждую длину принятого кода с символом в заранее определенном порядке. Например, длины кода могут быть извлечены в порядке 3, 4, 6, 2 и т.д. Затем процесс кодирования может связать каждую длину с соответствующим символом. Здесь порядок последовательный - (символ, длина)-(0,3) (1,4) (3,6) (4,2).

Как проиллюстрировано на фиг. 15, некоторые символы могут не отправляться, но флаг может указывать, что символ имеет соответствующую нулевую длину кода, то есть этот символ не существует в потоке битов, который должен быть декодирован (или не используется).

Затем каноническая операция кодирования Хаффмана может назначать код каждому символу на основе длины кода. Например, если длина кода равна 2, код для этого символа будет '1x'. Здесь x указывает, что значение является несущественным. В определенных реализациях код будет '10'.

Если длины идентичны, каноническая операция кодирования Хаффмана назначает коды на основе последовательного порядка символов. Каждый код назначается таким образом, что по мере анализа битового потока декодер может продолжать проверять следующий бит, пока не будет существовать только один возможный код. Например, первому последовательному символу длины 4 может быть назначен код 1110, а второму последовательному символу длины 4 может быть назначен код 1111. Таким образом, анализируя битовый поток, если первые биты битового потока равны 1110x, возможна лишь кодировка потоком битов первого последовательного символа длиной 4.

Соответственно, как только операция кодирования установила набор кодов, связанных с каждым символом, операция кодирования может перейти к синтаксическому анализу битового потока. В некоторых вариантах воплощения операция кодирования будет строить дерево, чтобы установить символ, связанный с кодом битового потока. То есть операция кодирования будет брать каждый бит битового потока и проходить по дереву в соответствии со значением бита в потоке битов, пока не будет найден лист. Как только лист найден, выводится символ, связанный с листом. Процесс продолжается в корне дерева

со следующим битом битового потока. Таким образом, операция кодирования может выводить набор символов, полученных из набора кодов, хранящихся в кодированном потоке битов. В предпочтительных вариантах воплощения каждый символ представляет собой байт, как описано в другом месте в данном документе.

Затем набор символов может быть передан в операцию кодирования длин серий. Операция кодирования длин серий идентифицирует тип символа и анализирует символ, чтобы извлечь значение данных или серию нулей. На основе значений данных и серий нулей операция может воссоздать исходные остаточные данные, которые были закодированы.

Как отмечалось выше в данном документе, выходной сигнал кодирования Хаффмана, вероятно, будет набором символов. Задача на следующем этапе декодирования состоит в том, чтобы извлечь из этих символов релевантную информацию, учитывая, что каждый символ содержит данные в разном формате, и каждый символ будет представлять различную информацию, несмотря на то, что информация не будет сразу различима из символа или байта.

В предпочтительных примерах операция кодирования обращается к конечному автомату, как проиллюстрировано на фиг. 18. В целом, операция кодирования сначала предполагает, что первый символ является типом значения данных (значение данных, конечно, может быть 0). Исходя из этого, операция кодирования может идентифицировать, включает ли байт флаг переполнения или флаг выполнения. Флаг переполнения и запуска описаны выше в документе и проиллюстрированы на фиг. 10.

Операция кодирования может сначала проверить младший значащий бит символа (или байта). В данном контексте, это флаг (или бит) переполнения. Если установлен младший значащий бит, операция кодирования определяет, что следующий символ также будет значением данных и будет иметь тип RLC_RESIDUAL_MSB. Остальные биты символа будут младшими значащими битами значения данных.

Процесс кодирования переходит к синтаксическому анализу следующего символа и извлечению младших битов как оставшейся части значения данных. Биты первого символа могут быть объединены с битами второго символа для восстановления значения данных.

В этом текущем символе, проиллюстрированном на фиг. 11, также будет флаг выполнения, здесь самый старший бит. Если это установлено, следующим символом будет символ запуска, а не символ данных.

В символе запуска, проиллюстрированном на фиг. 12, операция кодирования проверяет самый старший бит, чтобы указать, является ли следующий символ символом запуска или символом данных, и извлекает оставшиеся биты как серию нулей. То есть максимальную серию из 127 нулей.

Если символ данных указывает на отсутствие переполнения (во флаге переполнения, здесь наименее значимый бит), тогда символ также будет содержать в самом значительном бите бит выполнения, и операция кодирования сможет определить по этому биту, что следующий символ в наборе - это символ запуска или символ данных. Таким образом, операция кодирования может извлекать значение данных из битов 6-1. Здесь доступно 31 значение данных без переполнения.

Как уже отмечалось, конечный автомат, проиллюстрированный на фиг. 18, просто иллюстрирует процесс.

Начиная с символа RLC_RESIDUAL_LSB, если бит переполнения=0 и бит выполнения=0, то следующим символом будет символ RLC_RESIDUAL_LSB. Если бит переполнения=1, то следующим символом будет RLC_RESIDUAL_MSB. Если бит переполнения=1, бита выполнения не будет. Если бит выполнения=1 и бит переполнения=0, то следующим битом будет символ RLC_ZERO_RUN.

В символе RLC RLC_RESIDUAL_MSB, если бит выполнения равен 0, следующим символом будет символ RLC_RESIDUAL_LSB. Если бит выполнения равен 1, следующим символом будет символ RLC_ZERO_RUN.

В символе RLC_ZERO_RUN, если бит выполнения равен 0, следующим битом будет символ RLC_RESIDUAL_LSB. Если бит выполнения равен 1, следующим битом будет символ RLC_ZERO_RUN.

Разумеется, биты можно инвертировать (0/1, 1/0 и т.п.) без потери функциональности. Точно так же положения внутри символов или байтов флагов являются просто иллюстративными.

Операция кодирования длин серий может идентифицировать следующий символ в наборе символов и извлекать либо значение данных, либо серию нулей. Затем операция кодирования может объединить эти значения и нули для воссоздания остаточных данных. Порядок может быть в порядке извлечения или, альтернативно, в некотором заранее определенном порядке.

Таким образом, операция кодирования может выводить остаточные данные, которые были закодированы в байтовый поток.

Выше в данном документе было описано, как несколько метаданных или параметров кодирования могут использоваться в процессе кодирования. На стороне декодирования операция кодирования может включать в себя обратную связь между операциями. То есть ожидаемый следующий символ может быть извлечен из текущего символа (например, с использованием битов переполнения и выполнения), а ожидаемый следующий символ может быть передан операции кодирования Хаффмана.

Операция кодирования Хаффмана может необязательно генерировать отдельную кодовую книгу

для каждого типа символа (и извлекать несколько заголовков потока или строить таблицу из нескольких кодов и символов).

Предполагая, что первый символ будет значением данных, операция кодирования Хаффмана декодирует первый код для соответствия символу этого типа. Из согласованного символа операция кодирования может идентифицировать бит переполнения и/или бит выполнения и идентифицировать следующий тип символа. Затем операция кодирования Хаффмана может использовать соответствующую кодовую книгу для этого типа символа для извлечения следующего кода в потоке битов. Процесс переносится таким итеративным способом с использованием текущего декодированного символа для извлечения указания следующего символа и соответствующего изменения кодовых книг.

Выше в данном документе было описано, как операция декодирования может комбинировать операцию кодирования Хаффмана, предпочтительно операцию канонического кодирования Хаффмана, и операцию кодирования длин серий для воссоздания остаточных данных из закодированного битового потока. Следует понимать, что методы операции кодирования длин серий могут применяться к байтовому потоку, который не был закодирован с использованием операции кодирования Хаффмана. Точно так же операция кодирования Хаффмана может применяться без последующего этапа операции кодирования длин серий.

Ниже описывается пример конкретных этапов реализации декодера. В примере реализации, если поток состоит из более чем одной части, выполняются следующие шаги:

1. Считывание длин кода из заголовка потока (например, с помощью функции ReadCodeLengths):

а. установка длин кода для каждого символа;

б. Назначение кодов символам на основе длин кода (например, с помощью функции AssignCodes); и

в. генерирование таблицы для поиска подмножеств кодов одинаковой длины. Каждый элемент таблицы записывает первый индекс заданной длины и соответствующий код (firstldx, firstCode).

2. Декодирование данных RLE:

а. Присваивание контекста RLE символу RLC_RESIDUAL_LSB.

б. Декодирование текущего кода с поиском корректной длины кода в сгенерированной таблице и индексирование массива кодов с помощью: firstldx -(current_code - firstCode). Это работает, потому что все коды для заданной длины являются последовательными по построению дерева Хаффмана.

в. Использование конечного автомата RLE для получения следующего контекста (см. фиг. 18). Если поток не пустой, переход к б.

Декодер длин серии считывает данные, закодированные по длине серии, побайтно. По построению контекст первого байта данных гарантированно будет RLC_RESIDUAL_LSB. Декодер использует конечный автомат, проиллюстрированный на фиг. 18, для определения контекста следующего байта данных. Контекст сообщает декодеру, как интерпретировать текущий байт данных, как описано выше в данном документе.

Обратите внимание, что в этом конкретном примере реализации конечный автомат длины серий также используется процессом кодирования и декодирования Хаффмана, чтобы знать, какой код Хаффмана использовать для текущего символа или кодового слова.

Как в кодере, так и в декодере, например, реализованном в потоковом сервере или клиентском устройстве, или в клиентском устройстве, декодирующем данные из хранилища данных, способы и процессы, описанные в данном документе, могут быть воплощены в виде кода (например, программного кода) и/или данных. Кодер и декодер могут быть реализованы аппаратно или программно, что хорошо известно в области сжатия данных. Например, аппаратное ускорение с использованием специально запрограммированного графического процессора (GPU) или специально разработанной программируемой вентильной матрицы (FPGA) может обеспечить определенную эффективность. Для полноты такой код и данные могут храниться на одном или более машиночитаемых носителях, которые могут содержать любое устройство или носитель, который может хранить код и/или данные для использования компьютерной системой. Когда компьютерная система считывает и выполняет код и/или данные, хранящиеся на машиночитаемом носителе, компьютерная система выполняет способы и процессы, воплощенные в виде структур данных и кода, хранящихся на машиночитаемом носителе данных. В некоторых вариантах воплощения один или более этапов описанных здесь способов и процессов могут выполняться процессором (например, процессором компьютерной системы или системы хранения данных).

Как правило, любые функциональные возможности, описанные в этом тексте или проиллюстрированные на чертежах, могут быть реализованы с использованием программного обеспечения, встроенного программного обеспечения (например, схемы с фиксированными логическими функциями), программируемого или непрограммируемого оборудования или комбинации этих реализаций. Термины "компонент" или "функция", в контексте данного документа, обычно представляют программное обеспечение, встроенное программное обеспечение, аппаратное обеспечение или их комбинацию. Например, в случае программной реализации термины "компонент" или "функция" могут относиться к программному коду, который выполняет определенные задачи при выполнении на устройстве или устройствах обработки. Проиллюстрированное разделение компонентов и функций на отдельные блоки может отражать любую фактическую или концептуальную физическую группировку и распределение такого программного

обеспечения и/или оборудования и задач.

В настоящей заявке был описан способ кодирования и декодирования сигнала, в частности видеосигнала и/или сигнала изображения.

В частности, описан способ кодирования сигнала, включающий прием входного кадра и обработку входного кадра для генерации по меньшей мере одного первого набора остаточных данных, упомянутые остаточные данные позволяют декодеру восстановить исходный кадр из опорного восстановленного кадра.

В одном варианте воплощения способ включает получение восстановленного кадра из декодированного кадра, полученного из модуля декодирования, при этом модуль декодирования выполнен с возможностью генерирования упомянутого декодированного кадра путем декодирования первого кодированного кадра, который был закодирован согласно первому способу кодирования. Способ дополнительно содержит понижающую дискретизацию входного кадра для получения кадра с пониженной дискретизацией и передачу указанного кадра с пониженной дискретизацией в модуль кодирования, сконфигурированный для кодирования указанного кадра с пониженной дискретизацией в соответствии с первым способом кодирования для генерирования первого закодированного кадра. Получение восстановленного кадра может дополнительно включать повышающую дискретизацию декодированного кадра для генерирования восстановленного кадра.

В другом варианте воплощения способ включает получение восстановленного кадра из комбинации второго набора остаточных данных и декодированного кадра, полученного из модуля декодирования, при этом модуль декодирования выполнен с возможностью генерирования указанного декодированного кадра путем декодирования первого кодированного кадра, который был закодирован согласно первому способу кодирования. Способ дополнительно включает понижающую дискретизацию входного кадра для получения кадра с пониженной дискретизацией и передачу указанного кадра с пониженной дискретизацией в модуль кодирования, выполненный с возможностью кодирования указанного кадра с пониженной дискретизацией в соответствии с первым способом кодирования, чтобы сгенерировать первый закодированный кадр. Способ дополнительно включает формирование упомянутого второго набора остаточных данных путем взятия разницы между декодированным кадром и кадром с пониженной дискретизацией. Способ дополнительно включает кодирование упомянутого второго набора остаточных данных для создания первого набора закодированных остаточных данных. Кодирование упомянутого второго набора остаточных данных может выполняться согласно второму способу кодирования. Второй способ кодирования включает преобразование второго набора остаточных данных в преобразованный второй набор остаточных данных. Преобразование второго набора остаточных данных включает выбор подмножества второго набора остаточных данных и применение преобразования к упомянутому подмножеству для создания соответствующего подмножества преобразованного второго набора остаточных данных. Один из подмножества преобразованного второго набора остаточных данных может быть получен путем усреднения подмножества второго набора остаточных данных. Получение восстановленного кадра может дополнительно включать повышающую дискретизацию комбинации второго набора остаточных данных и декодированного кадра для генерирования восстановленного кадра.

В одном варианте воплощения генерирование по меньшей мере одного набора остаточных данных включает взятие разницы между опорным восстановленным кадром и входным кадром. Способ дополнительно включает кодирование упомянутого первого набора остаточных данных для создания первого набора кодированных остаточных данных. Кодирование упомянутого первого набора остаточных данных может выполняться согласно третьему способу кодирования. Третий способ кодирования включает преобразование первого набора остаточных данных в преобразованный первый набор остаточных данных. Преобразование первого набора остаточных данных включает выбор подмножества первого набора остаточных данных и применение преобразования к упомянутому подмножеству для генерации соответствующего подмножества преобразованного первого набора остаточных данных. Один из подмножеств преобразованного первого набора остаточных данных может быть получен посредством разности между средним значением поднабора входного кадра и соответствующего элемента комбинации второго набора остаточных данных и декодированного кадра.

В частности, описан способ декодирования сигнала, содержащий прием кодированного кадра и по меньшей мере одного набора кодированных остаточных данных. Первый кодированный кадр может быть закодирован с использованием первого способа кодирования.

По меньшей мере один набор остаточных данных может быть закодирован с использованием второго и/или третьего способа кодирования.

Способ дополнительно включает передачу первого кодированного кадра в модуль декодирования, при этом модуль декодирования выполнен с возможностью генерирования декодированного кадра путем декодирования кодированного кадра, который был закодирован согласно первому способу кодирования.

Способ может дополнительно включать декодирование по меньшей мере одного набора закодированных остаточных данных согласно соответствующему способу кодирования, используемому для их кодирования.

В одном варианте воплощения первый набор кодированных остаточных данных декодируется пу-

тем применения второго способа декодирования, соответствующего упомянутому второму способу кодирования, для получения первого набора декодированных остаточных данных. Способ дополнительно содержит объединение первого набора остаточных данных с декодированным кадром для получения объединенного кадра. Способ дополнительно содержит повышающую дискретизацию комбинированного кадра, чтобы получить опорный декодированный кадр.

Способ дополнительно включает декодирование второго набора кодированных остаточных данных путем применения третьего способа декодирования, соответствующего упомянутому третьему способу кодирования, для получения второго набора декодированных остаточных данных. Способ дополнительно включает объединение второго набора декодированных остаточных данных с опорным декодированным кадром для получения восстановленного кадра.

В другом варианте воплощения способ включает повышение дискретизации декодированного кадра, чтобы получить опорный декодированный кадр.

Способ дополнительно включает декодирование набора закодированных остаточных данных путем применения второго или третьего способа декодирования, соответствующего упомянутому второму или третьему способу кодирования, для получения набора декодированных остаточных данных. Способ дополнительно содержит объединение набора декодированных остаточных данных с опорным декодированным кадром для получения восстановленного кадра.

Следующие ниже утверждения описывают предпочтительные или иллюстративные аспекты, описанные и проиллюстрированные в данном документе.

Способ кодирования входного видео во множество кодированных потоков, так что кодированные потоки могут быть объединены для восстановления входного видео, способ, включающий:

получение входного видео с полным разрешением;

понижение дискретизации входного видео с полным разрешением для создания видео с пониженной дискретизацией;

кодирование видео с пониженной дискретизацией с использованием первого кодека для создания базового кодированного потока;

восстановление видео из кодированного видео для создания восстановленного видео;

сравнение восстановленного видео с исходным видео и создание одного или более дополнительно закодированных потоков на основе сравнения.

Входное видео по сравнению с восстановленным видео может быть видео с пониженной дискретизацией.

Согласно одному из примеров способа, сравнение восстановленного видео с входным видео включает: сравнение восстановленного видео с видео, имеющим пониженную дискретизацию, для создания первого набора остаточных данных, и при этом создание одного или более дополнительных кодированных потоков включает кодирование первого набора остаточных данных для создания кодированного потока первого уровня.

Входное видео по сравнению с восстановленным видео может быть входным видео с полным разрешением, а восстановленное видео может быть подвергнуто повышению дискретизации.

Согласно одному из примеров способа, сравнение восстановленного видео с входным видео включает: повышение дискретизации восстановленного видео для создания восстановленного видео с повышенной дискретизацией и сравнение восстановленного видео с повышенной дискретизацией и входного видео с полным разрешением для создания второго набора остаточных данных, и при этом создание одного или более дополнительных кодированных потоков включает кодирование второй разности для создания кодированного потока второго уровня.

Соответственно, в одном из примеров, способ предусматривает генерирование базового кодированного потока, кодированного потока первого уровня и кодированного потока второго уровня согласно описанным выше примерам способов. Каждый из кодированного потока первого уровня и кодированного потока второго уровня может содержать данные расширения, используемые декодером для расширения кодированного базового потока.

Остаточные данные могут выступать в качестве различий между двумя видео или кадрами.

Кодированные потоки могут сопровождаться одним или более заголовками, которые содержат параметры, указывающие аспекты процесса кодирования для облегчения декодирования. Например, заголовки могут содержать используемый кодек, применяемое преобразование, применяемое квантование и/или другие параметры декодирования.

Выше в документе было описано, как этап ранжирования и выбора может применяться к остаточным данным, может выполняться этап вычитания временных коэффициентов, а также может быть адаптировано квантование. Каждый из этих шагов может быть заранее определен и выборочно применен или может применяться на основе анализа входящего видео, видео с пониженной дискретизацией, восстановленного видео, видео с повышенной дискретизацией или любой комбинации вышеперечисленного для улучшения общей производительности кодера. Этапы могут применяться выборочно на основе заранее определенного набора правил или применяться непосредственно на основе анализа или обратной связи производительности.

Один из примеров способа дополнительно включает: отправку базового закодированного потока. Один из примеров способа дополнительно включает: отправку закодированного потока первого уровня. Один из примеров способа дополнительно включает: отправку закодированного потока второго уровня.

Согласно дополнительному аспекту настоящего изобретения предоставляется способ декодирования.

Способ декодирования множества закодированных потоков в восстановленное выходное видео, включающий:

прием первого базового кодированного потока;

декодирование первого базового кодированного потока согласно первому кодеку для генерации первого выходного видео;

получение одного или более дополнительно закодированных потоков;

декодирование одного или более дополнительно закодированных потоков для генерирования набора остаточных данных и объединение набора остаточных данных с первым видео для создания декодированного видео.

В одном примере способ содержит извлечение множества параметров декодирования из заголовка. Параметры декодирования могут указывать, какие процедурные этапы были включены в процесс кодирования.

В одном примере способ может включать прием закодированного потока первого уровня и прием закодированного потока второго уровня. В этом примере этап декодирования одного или более дополнительно закодированных потоков для генерации набора остаточных данных включает:

декодирование закодированного потока первого уровня для получения первого набора остаточных данных;

при этом этап комбинирования набора остаточных данных с первым видео для генерирования декодированного видео включает:

объединение первого набора остаточных данных с первым выходным видео для генерации второго выходного видео;

повышающая дискретизация второго выходного видео для создания второго выходного видео с повышенной дискретизацией;

декодирование закодированного потока второго уровня для получения второго набора остаточных данных и

объединение второго набора остаточных данных со вторым выходным видео для создания восстановленного выходного видео.

Способ может дополнительно включать отображение или вывод восстановленного вывода.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Способ кодирования видеосигнала, включающий: получение входного кадра (100);

обработку входного кадра (100) для генерирования остаточных данных (110, 119), причем остаточные данные формируют часть расширенного потока и позволяют декодеру восстановить входной кадр на основе опорного восстановленного кадра;

применение операции кодирования длин серий к остаточным данным (110, 119),

при этом операция кодирования длин серий включает генерирование байтового потока, закодированного по длинам серий, содержащего набор символов, представляющих ненулевые значения данных остаточных данных, и подсчеты последовательных нулевых значений остаточных данных; и

причем способ характеризуется следующими этапами:

применение канонической операции кодирования Хаффмана к набору символов для генерирования данных, закодированных по Хаффману, содержащих набор кодов, представляющих байтовый поток, закодированный по длине серий;

сравнение размера данных по меньшей мере части байтового потока, закодированного по длине серий, с размером данных по меньшей мере соответствующей части данных, закодированных по Хаффману; и

вывод байтового потока, закодированного по длине серий, или данных, закодированных по Хаффману, в выходной битовый поток (102, 103) на основе того, какой из них имеет меньший размер данных.

2. Способ по п.1, в котором операция кодирования длин серий включает: кодирование ненулевых значений данных остаточных данных по меньшей мере в первый тип символа и кодирование подсчетов последовательных нулевых значений во второй тип символа, так что остаточные данные кодируются как последовательность символов разных типов.

3. Способ по п.1 или п.2, в котором операция кодирования длин серий включает: кодирование значений данных остаточных данных в первый тип символа и третий тип символа, причем первый и третий типы символа каждый содержит часть значения данных, так что части могут быть объединены в декодере для восстановления значений данных.

4. Способ по п.3, в котором операция кодирования длин серий включает: сравнение размера каждого значения данных остаточных данных, подлежащих кодированию, с пороговым значением и кодирова-

ние каждого значения данных в первый тип символа, если размер ниже порогового значения, и кодирование части каждого значения данных в первый тип символа и части каждого значения данных в третий тип символа, если размер выше порогового значения.

5. Способ по п.4, который дополнительно включает: если размер превышает пороговое значение, установку флага в символе первого типа символа, указывающего на часть представленного значения данных, кодируют в дополнительный символ третьего типа символа.

6. Способ по любому из пп.2-5, который дополнительно включает: вставку флага в каждый символ, указывающий тип символа, кодируемого следующим в байтовом потоке, закодированном по длине серий.

7. Способ по любому из предшествующих пунктов, который дополнительно включает: добавление флага к метаданным конфигурации, сопровождающим выходной битовый поток, указывающего, представляет ли битовый поток байтовый поток, закодированный по длине серий, или данные, закодированные по Хаффману.

8. Способ по любому из предшествующих пунктов, в котором операция кодирования Хаффмана включает: формирование отдельной таблицы частот для символов первого типа символа и символов второго типа символа.

9. Способ по любому предшествующих пунктов, который дополнительно включает: генерирование заголовка потока, содержащего указание множества длин кодов, так что декодер может выводить длины кодов и соответствующие символы для операции канонического декодирования Хаффмана.

10. Способ по п.9, в котором заголовок потока является одним из следующего: первым типом заголовка потока, содержащим указание символа, связанного с соответствующей длиной кода из множества длин кода.

11. Способ по п.9, в котором заголовок потока является вторым типом заголовка потока, причем для второго типа заголовка потока способ дополнительно включает упорядочивание множества длин кода в заголовке потока на основе заранее определенного порядка символов, соответствующих каждой из длин кода, так что длины кода могут быть связаны с соответствующим символом в декодере.

12. Способ по п.11, в котором второй тип заголовка потока дополнительно содержит флаг, указывающий, что символ в заранее определенном порядке возможного набора символов не существует в наборе символов в байтовом потоке, закодированном по длине серий.

13. Способ по любому из пп.10-12, который дополнительно включает: сравнение количества уникальных кодов в данных, закодированных по Хаффману, с пороговым значением и создание первого типа заголовка потока или второго типа заголовка потока на основе сравнения.

14. Способ декодирования видеосигнала, включающий:

получение закодированного битового потока (102, 103);

декодирование закодированного битового потока (102, 103) для генерирования остаточных данных для расширенного потока и

восстановление исходного кадра видеосигнала из остаточных данных и опорного восстановленного кадра,

причем этап декодирования закодированного битового потока (102, 103) включает:

применение операции кодирования длин серий для генерирования остаточных данных, при этом операция кодирования длин серий включает:

идентификацию набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений остаточных данных;

анализ набора символов для получения ненулевых значений данных остаточных данных и подсчета последовательных нулевых значений и

генерирование остаточных данных из ненулевых значений данных и подсчетов последовательных нулевых значений,

причем способ характеризуется тем, что этап декодирования закодированного битового потока включает:

извлечение флага из метаданных конфигурации, сопровождающих закодированный битовый поток (102. 103), указывающего, содержит ли закодированный битовый поток байтовый поток, закодированный по длине серии, или данные, закодированные по Хаффману; и

выборочное применение канонической операции кодирования Хаффмана к закодированному битовому потоку (102, 103) на основе флага для генерирования набора символов, представляющих ненулевые значения данных остаточных данных, и набора символов, представляющих подсчеты последовательных нулевых значений остаточных данных, при этом этап применения операции кодирования длин серий для генерирования остаточных данных выполняют над наборами символов.

15. Способ по п.14, в котором операция кодирования длин серий включает:

идентификацию символов первого типа символа, представляющих ненулевые значения данных остаточных данных;

идентификацию символов второго типа символа, представляющих подсчеты последовательных нулевых значений, так что последовательность символов разных типов декодируется для генерации оста-

точных данных; и

анализ набора символов согласно соответствующему типу каждого символа.

16. Способ по п.14 или п.15, в котором операция кодирования длин серий включает:

идентификацию символов первого типа символа и символов третьего типа символа, причем каждый первый и третий типы символа представляют часть значения данных;

синтаксический анализ символа первого типа символа и символа третьего типа символа для получения частей значения данных и

объединение полученных частей значения данных в значение данных.

17. Способ по п.16, который дополнительно включает: извлечение флага переполнения из символа первого типа символа, указывающего, содержится ли часть значения данных символа первого типа символа в следующем третьем типе символа.

18. Способ по любому из пп.15-17, который дополнительно включает: извлечение флага из каждого символа, указывающего следующий тип символа, который ожидается в наборе символов.

19. Способ по любому из пп.14-18, который дополнительно включает:

извлечение заголовка потока, содержащего указание множества длин кода, которые должны использоваться для операции канонического кодирования Хаффмана;

связывание каждой длины кода с соответствующим символом в соответствии с канонической операцией кодирования Хаффмана и

идентификацию кода, связанного с каждым символом, на основе длин кода в соответствии с канонической операцией кодирования Хаффмана.

20. Способ по п.19, в котором заголовок потока содержит указание символа, связанного с соответствующей длиной кода из множества длин кода, а этап связывания каждой длины кода с символом включает связывание каждой длины кода с соответствующим символом в заголовке потока.

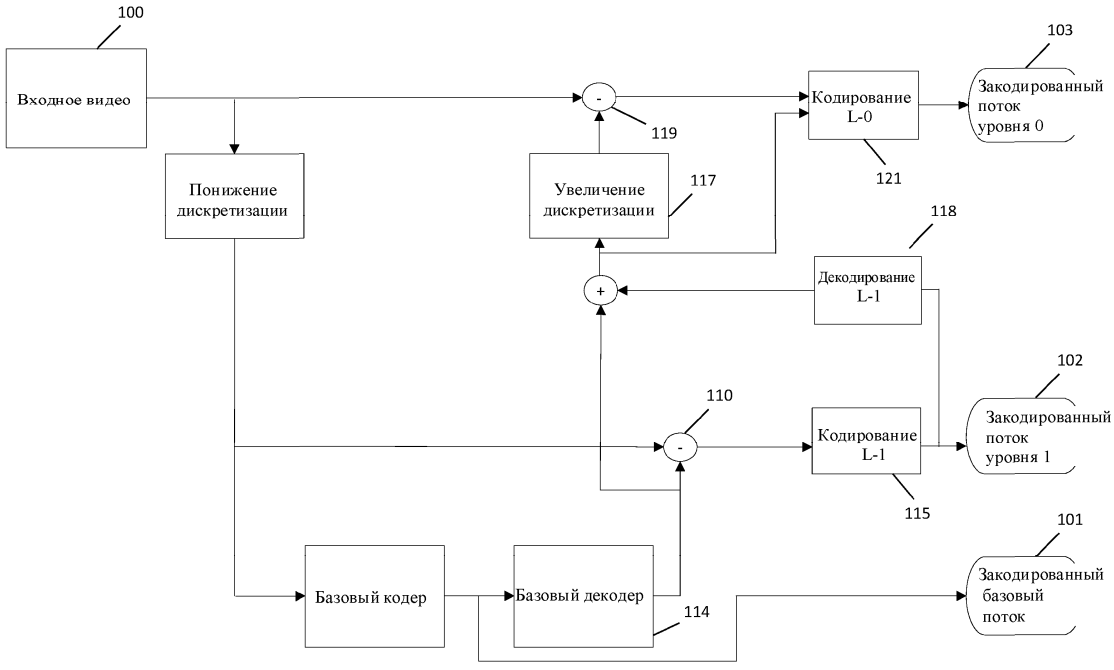
21. Способ по п.19, в котором этап связывания каждой длины кода с соответствующим символом включает связывание каждой длины кода с символом из набора заранее определенных символов в соответствии с порядком, в котором была извлечена каждая длина кода.

22. Способ по п.21, дополнительно включающий этап, на котором символ, из набора заранее определенных символов, не связывают с соответствующей длиной кода, когда флаг заголовка потока указывает, что длина кода не присутствует в заголовке потока для этого символа.

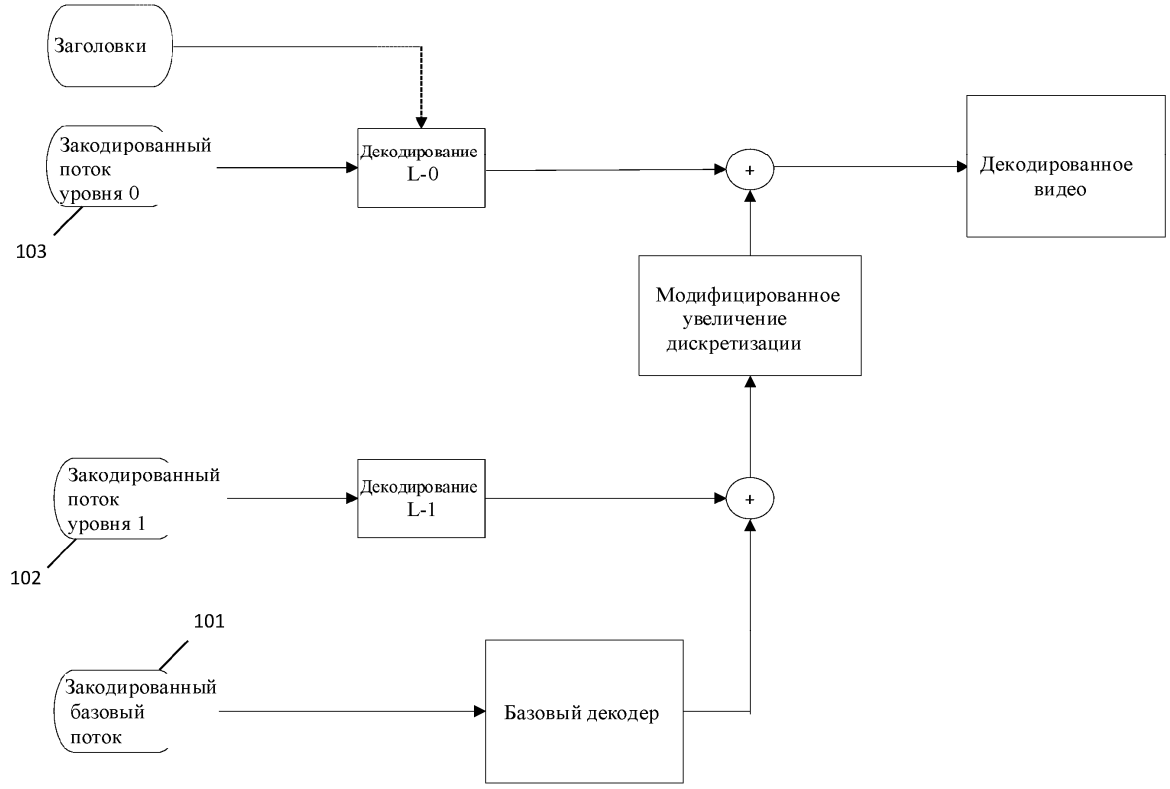
23. Устройство кодирования, выполненное с возможностью выполнения способа по любому из пп.1-13.

24. Устройство декодирования, выполненное с возможностью выполнения способа по любому из пп.14-22.

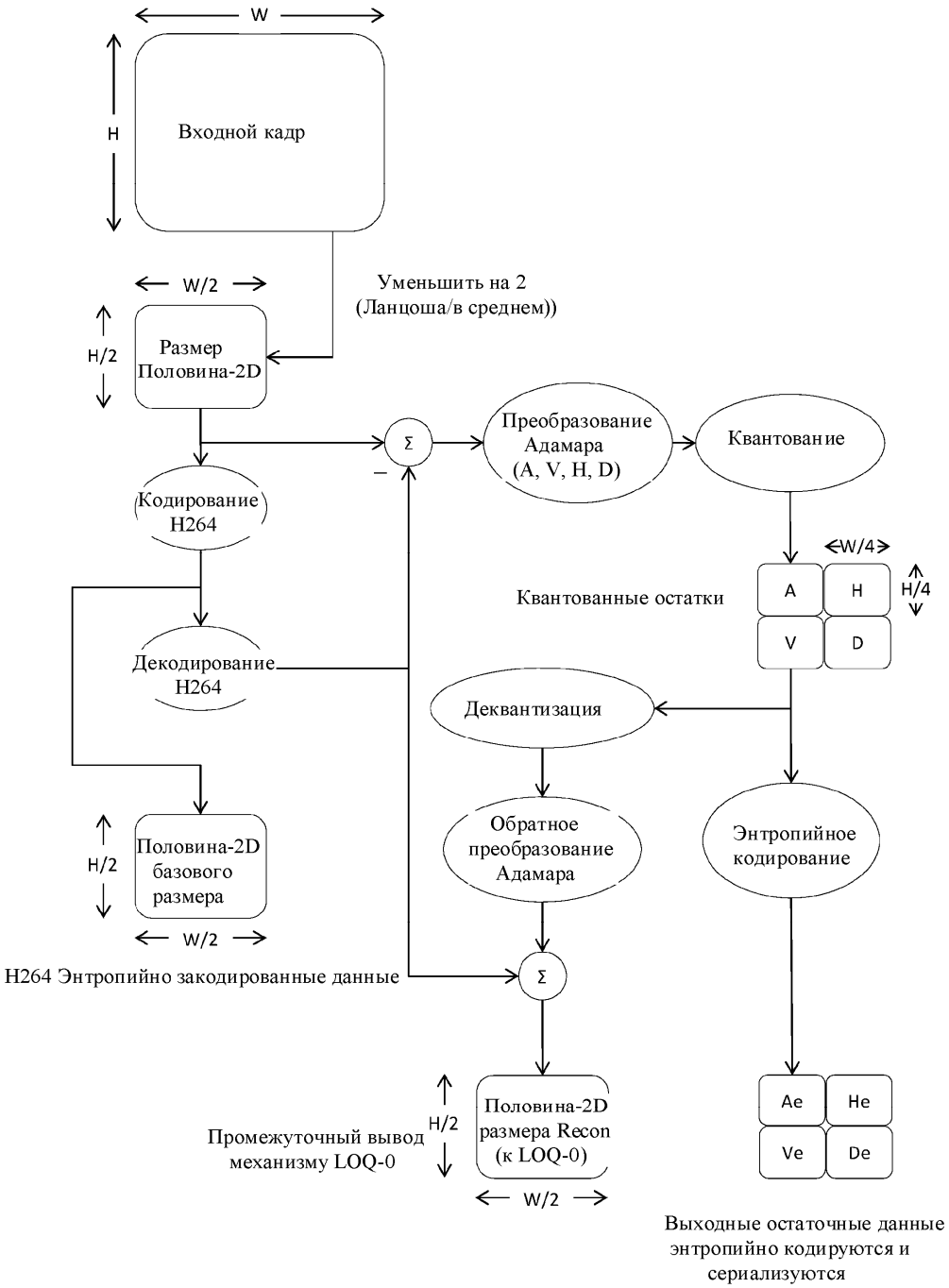
25. Машиночитаемый носитель, содержащий инструкции, которые при выполнении процессором побуждают процессор выполнять способ по любому из пп.1-22.



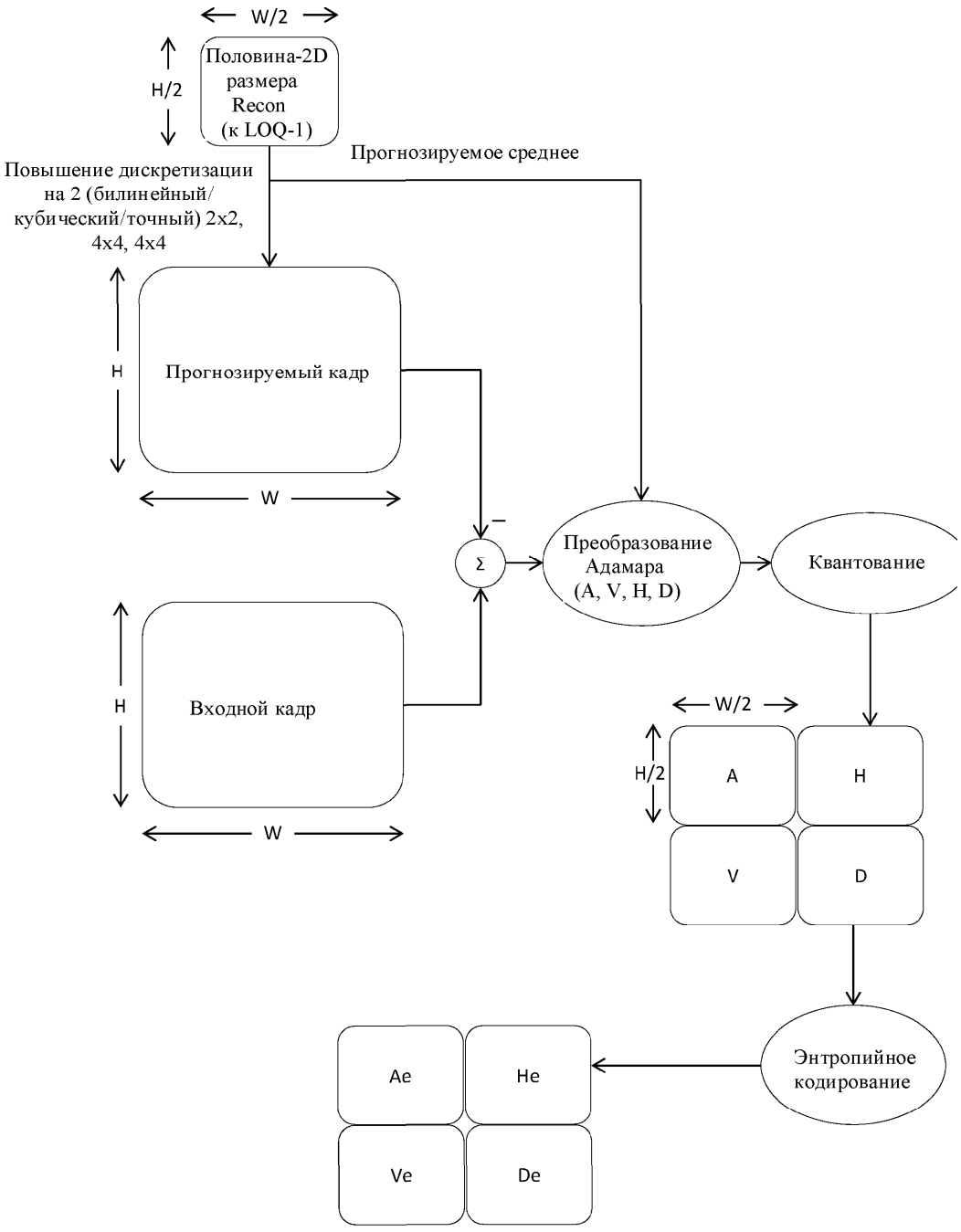
Фиг. 1


Заголовки
Закодированный поток уровня 0
103
Декодирование L-0
+
Декодированное видео
Модифицированное увеличение дискретизации
Закодированный поток уровня 1
102
Декодирование L-1
+
101
Закодированный базовый поток
Базовый декодер

Фиг. 2

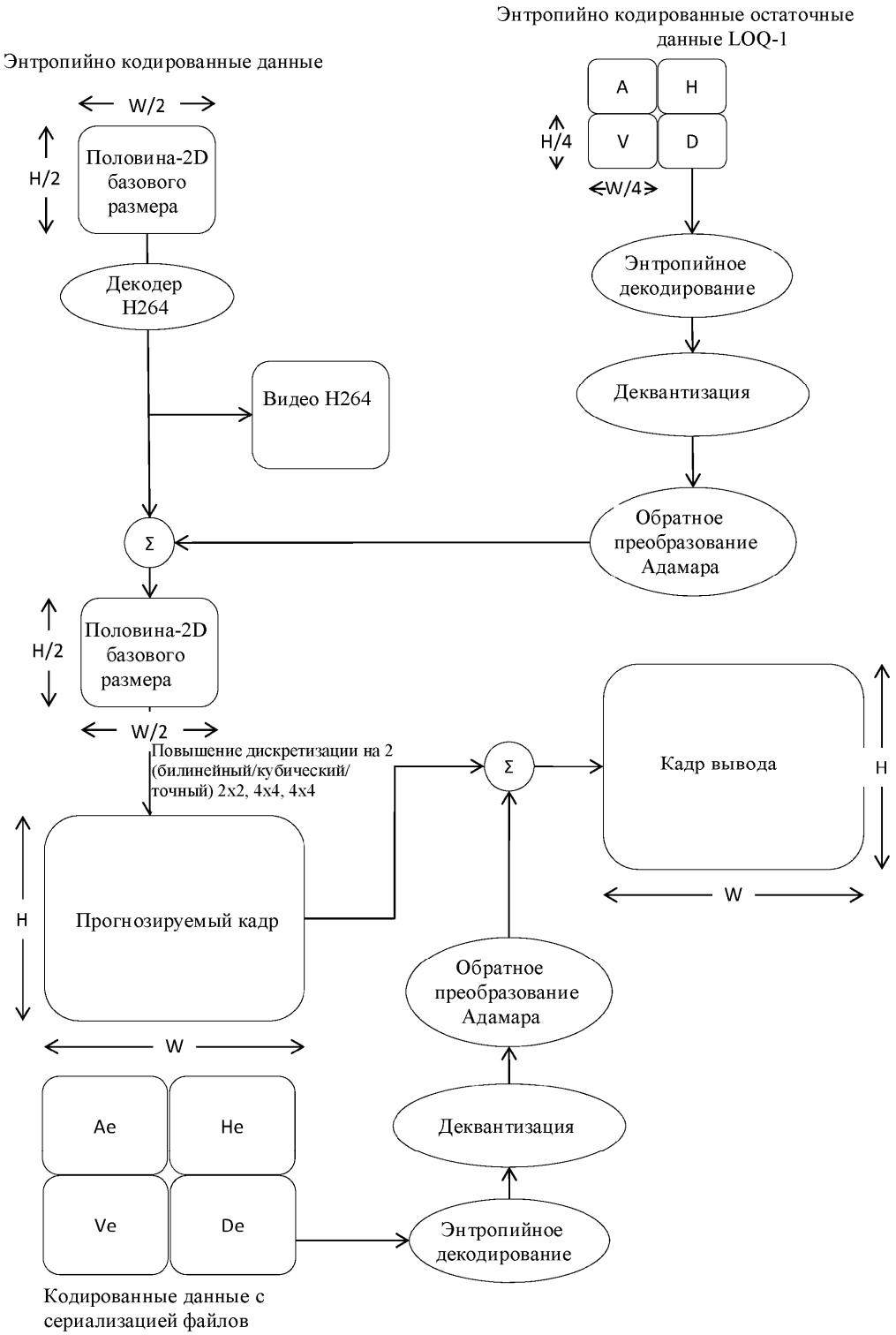

W
H
Входной кадр
W/2
H/2
Размер
Половина-2D
Уменьшить на 2
(Ланцоша/в среднем))
Σ
−
Преобразование
Адамара
(A, V, H, D)
Квантование
Кодирование
H264
W/4
H/4
A
H
V
D
Квантованные остатки
Декодирование
H264
Деквантизация
Обратное
преобразование
Адамара
Энтропийное
кодирование
H/2
Половина-2D
базового
размера
W/2
H264 Энтропийно закодированные данные
Σ
Промежуточный вывод
механизму LOQ-0
H/2
Половина-2D
размера Recon
(к LOQ-0)
W/2
Ae
He
Ve
De
Выходные остаточные данные
энтропийно кодируются и
сериализуются

Фиг. 3

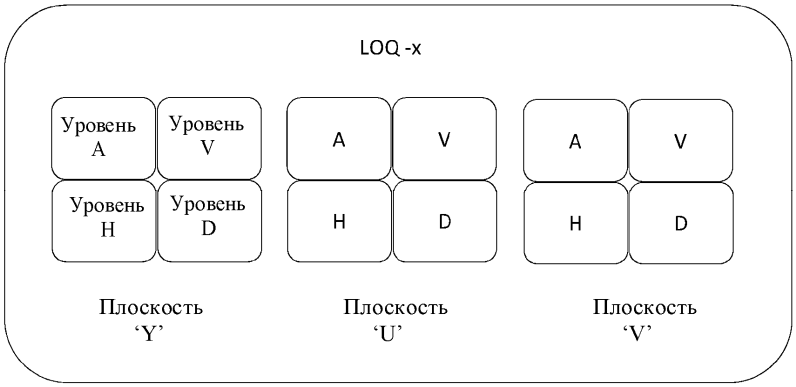


Выходные остаточные данные энтропийно кодируются и сериализуются

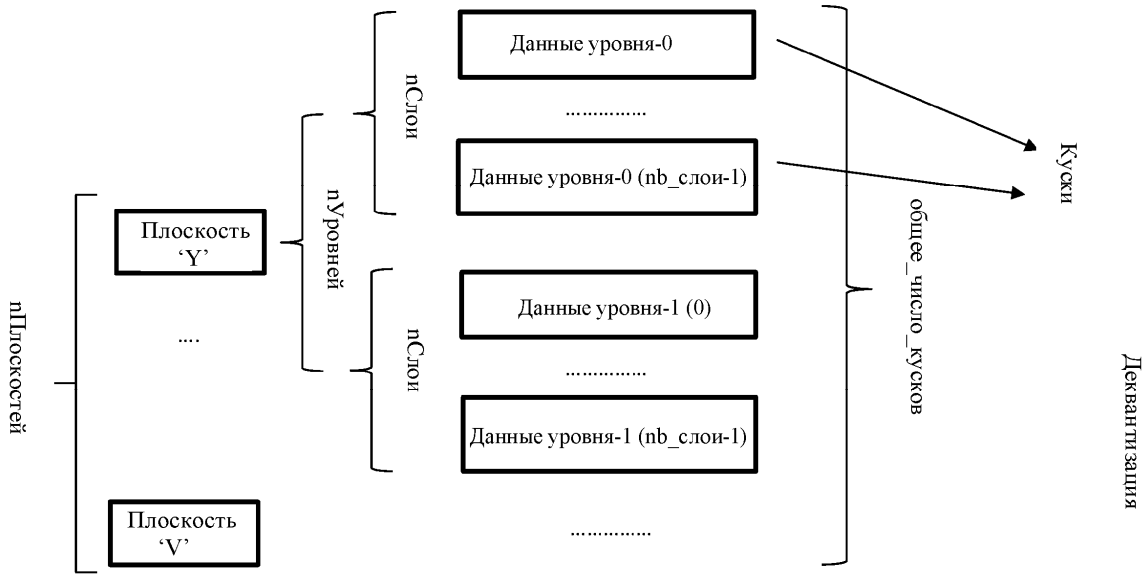
Фиг. 4


Энтропийно кодированные данные
Энтропийно кодированные остаточные данные LOQ-1
W/2
H/2
Половина-2D базового размера
Декодер H264
Видео H264
A
H
V
D
H/4
W/4
Энтропийное декодирование
Деквантизация
Обратное преобразование Адамара
Σ
H/2
Половина-2D базового размера
W/2
Повышение дискретизации на 2 (билинейный/кубический/ точный) 2x2, 4x4, 4x4
Σ
Кадр вывода
H
W
H
Прогнозируемый кадр
W
Обратное преобразование Адамара
Ae
He
Деквантизация
Ve
De
Энтропийное декодирование
Кодированные данные с сериализацией файлов

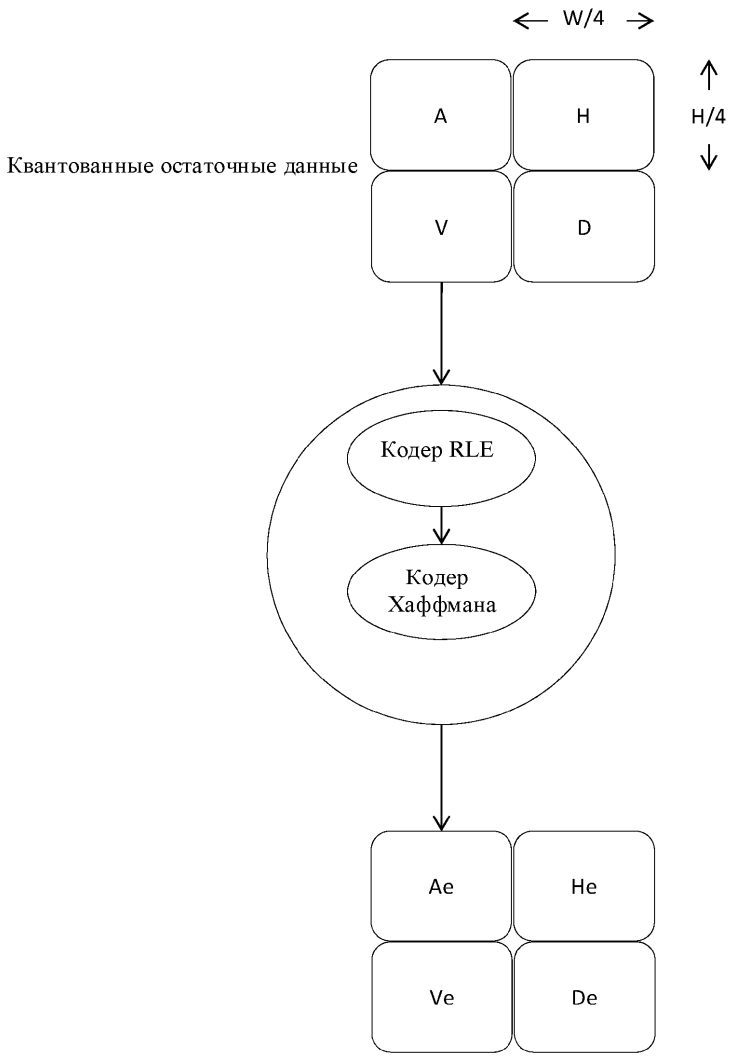
Фиг. 5


LOQ -x
Уровень A
Уровень V
Уровень H
Уровень D
A
V
H
D
A
V
H
D
Плоскость 'Y'
Плоскость 'U'
Плоскость 'V'

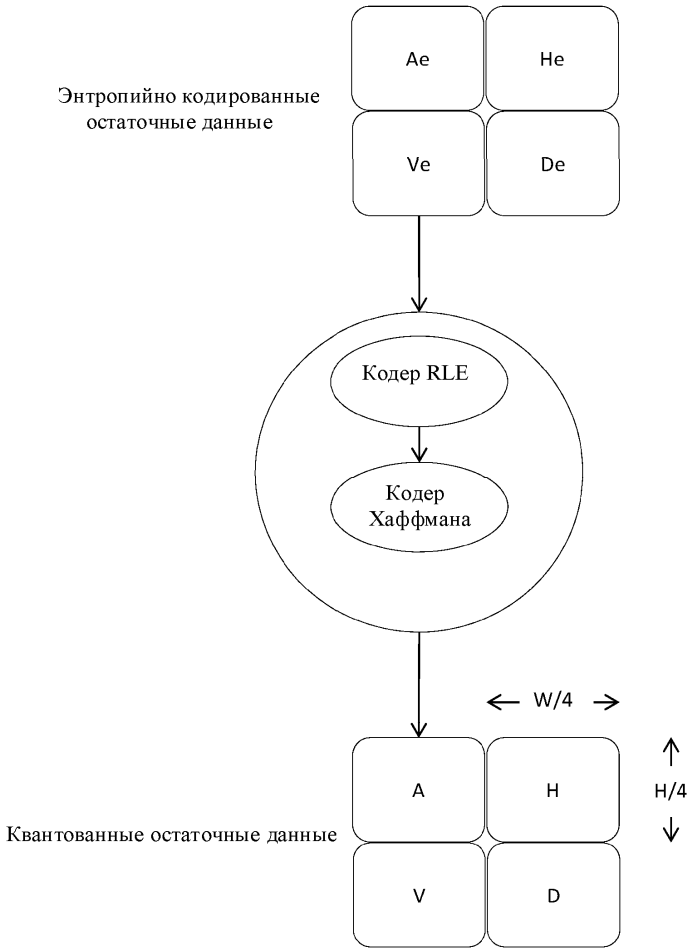
Фиг. 6


nПлоскостей
Плоскость 'Y'
....
Плоскость 'V'
nУровней
nСлои
nСлои
Данные уровня-0
.....
Данные уровня-0 (nb_слои-1)
Данные уровня-1 (0)
.....
Данные уровня-1 (nb_слои-1)
.....
общее_число_кусков
Куски
Деквантизация

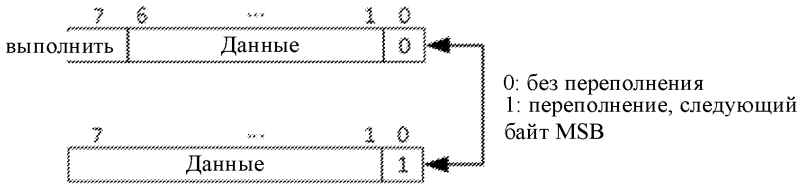
Фиг. 7


W/4
H/4
A
H
V
D
Квантованные остаточные данные
Кодер RLE
Кодер Хаффмана
Ae
He
Ve
De
Энтропийно кодированные остаточные данные

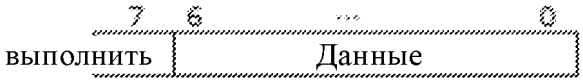
Фиг. 8



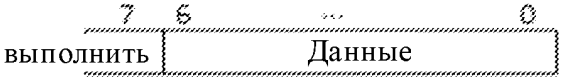
Фиг. 9



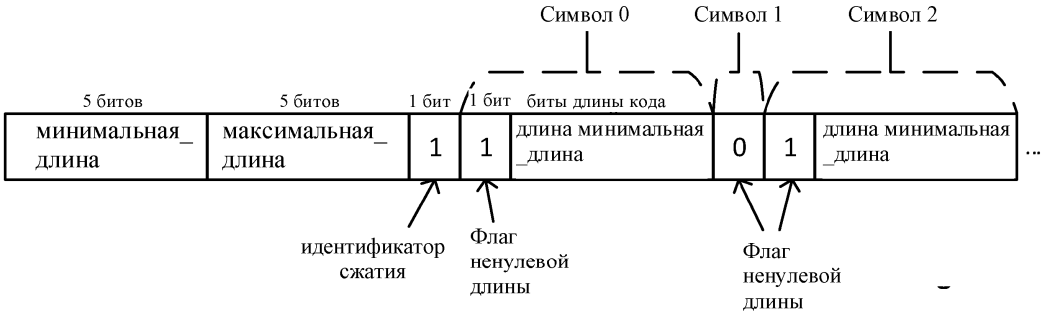
Фиг. 10



Фиг. 11

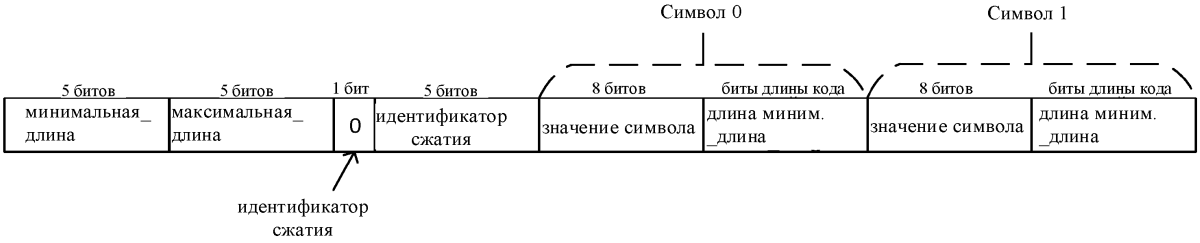


Фиг. 12



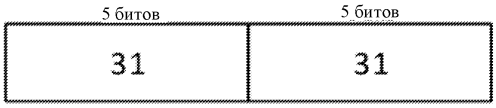
Длина кода в битах = log2(max_length-min_length+1)

Фиг. 13

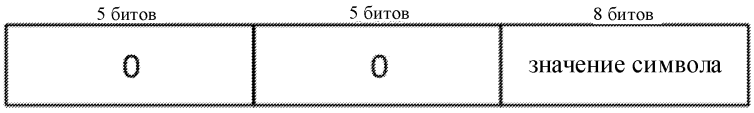


Длина кода в битах = log2(max_length-min_length+1)

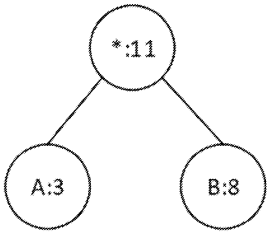
Фиг. 14



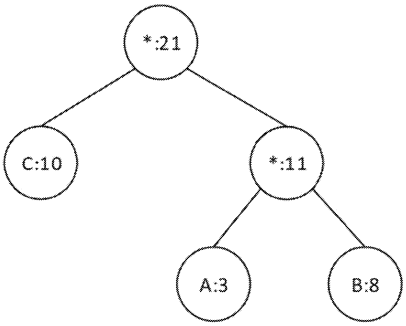
Фиг. 15



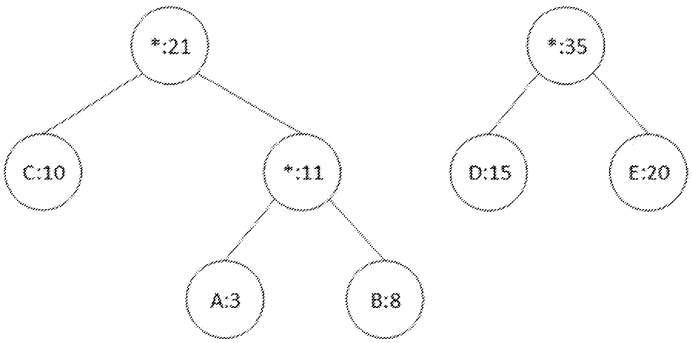
Фиг. 16



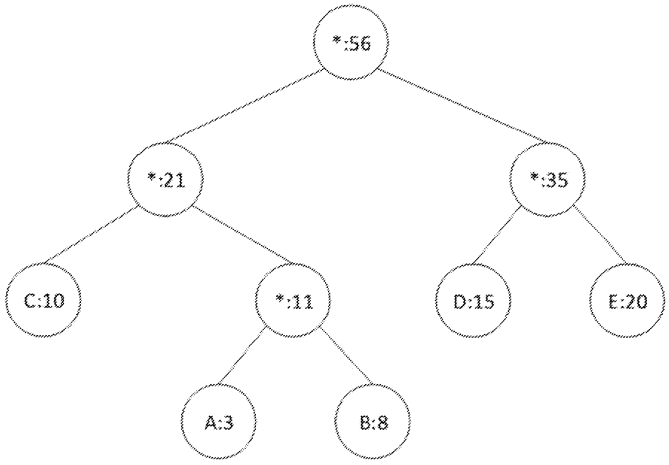
Фиг. 17a



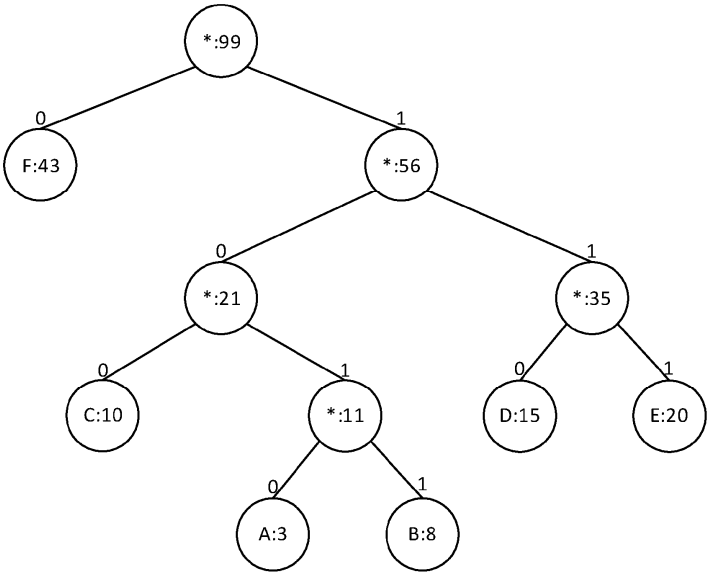
Фиг. 17b



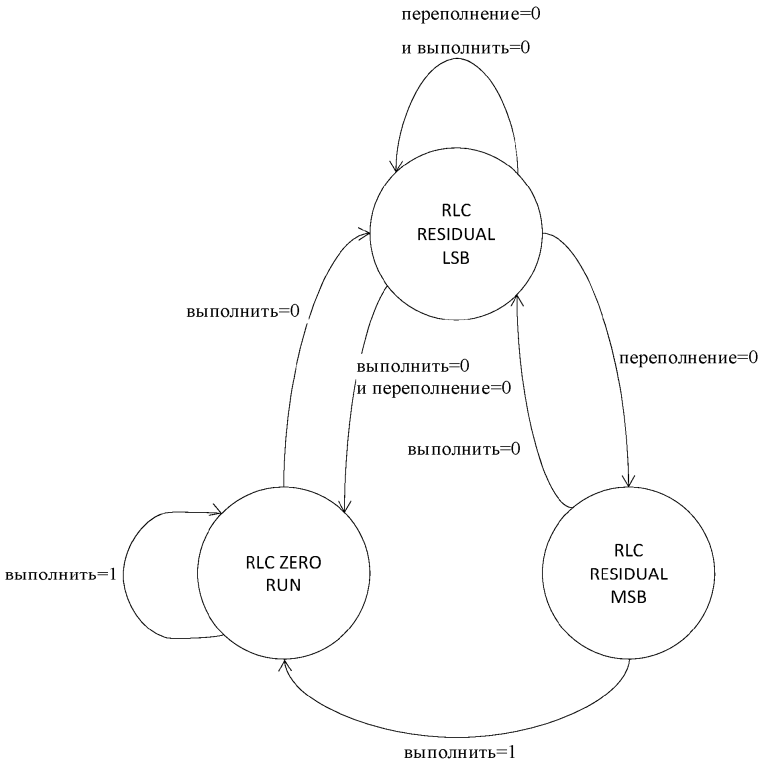
Фиг. 17c



Фиг. 17d



Фиг. 17e


переполнение=0
и выполнить=0
RLC
RESIDUAL
LSB
выполнить=0
выполнить=0
и переполнение=0
переполнение=0
выполнить=0
RLC ZERO
RUN
RLC
RESIDUAL
MSB
выполнить=1
выполнить=1

Фиг. 18