Реферат

В данной работе представлено описание разработки метода сжатия видео.

Цель работы: разработка метода сжатия видео на основе кратномасштабного анализа.

В дипломной работе приводится анализ существующих алгоритмов сжатия видео. Выделяются особенности существующих алгоритмов, их достоинства и недостатки. Приводится сравнение существующих алгоритмов сжатия видео.

Был предложен метод сжатия видео на основе кратномасштабного анализа, разработан алгоритм на основе предложенного метода. Также был создан программный комплекс, позволяющий применять разработанный алгоритм к видео.

В работе было проведено исследование исследование работы предложенного метода. Были определены типы видео, применение метода к которым дает наиболее эффективное сжатие видео.

Содержание

Глоссарий

Вейвлетное преобразование (англ. wavelet transform) — Инструмент, разбивающий данные, или функции, или операторы на составляющие с разными частотами, каждая из которых затем изучается с разрешением, подходящим по масштабу [1]

Битрейт — Величина потока данных, передаваемого в реальном времени (минимальный размер канала, который сможет пропустить этот поток без задержек). Частный случай — битрейт сжатого звука или видео.

Класс изображений/видео — совокупность изображений/видео, применение к которым алгоритма архивации дает качественно одинаковые результаты [2]

Коэффициент сжатия — отношение длины сжатых данных к длине соответствующих им несжатых данных [2]

Степень сжатия — отношение длины несжатых данных к длине соответствующих им сжатых данных [2]

Порядковая статистика (order statistic) — i-я порядковая статистика множества, состоящего из n элементов – это i-й элемент в порядке возрастания [3]

Цифровое видео — последовательность кадров, в которой каждый кадр рассматривается как набор отсчетов аналогового изображения. Отсчеты - пиксели [4]

Обозначения и сокращения

CWT — continuous wavelet transform, непрерывное вейвлетное преобразование

DWT — discrete wavelet transform, дискретное вейвлетное преобразование

YUV — цветовая модель, в которой цвет представляется как 3 компоненты — яркость (Y) и две цветоразностных (U и V)

YDbDr — цветовая модель, в которой цвет представляется как 3 компоненты — яркость (Y) и две цветоразностных (Db и Dr)

YIQ — цветовая модель, в которой цвет представляется как 3 компоненты — яркость (Y) и две цветоразностных (I и Q)

RGB (ARGB) — аддитивная цветовая модель, в которой цвет представляется как 3 компоненты — красная (R), синяя (B) и зеленая (G), значение которых добавляется к значению черного цвета. Возможно наличие компоненты A - альфа канала

HSV — цветовая модель, в которой цвет представляется как 3 компоненты - Hue — цветовой тон, Saturation — насыщенность, Value (значение цвета) или Brightness — яркость

CMYK — цветовая модель, в которой цвет представляется компонентами: C - Cyan, M - Magenta, Y - Yellow, B - Black

PSNR — пиковое отношение сигнала к шуму - соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала

SSIM — индекс структурного сходства (от англ. structure similarity) - один из методов измерения схожести между двумя изображениями

Введение

Развитие интернета, рост производительности компьютеров и прогресс в технологии производства цифровых камер, сканеров и принтеров привели к широкому использованию цифровых данных, в том числе видео. В последнее время наблюдается бурное развитие телекоммуникационных систем, предназначенных для приема и передачи видеоданных. Для хранения видео информации требуется больший объем, чем для других типов данных, таких как звуковая, текстовая информация или изображения. С ростом разрешающей способности экранов современных персональных компьютеров, а так же экранов кинотеатров качество видео и их размер постоянно растут. Размер графических данных файла с видео пропорционален количеству кадров, количеству пикселей в каждом кадре и количеству битов, требуемых для представления глубины цвета каждого пикселя. Без использования алгоритмов сжатия файл с графическими данными может занимать объем, сопоставимый с емкостью носителей современных персональных компьютеров. Необходимо улучшать алгоритмы сжатия данных, представляющих цифровой поток видеоданных. Сжатие данных важно как для скорости передачи, так и для эффективности хранения [5].

В настоящее время разработаны алгоритмы сжатия без потерь на основе универсальных методов сжатия и алгоритмы сжатия с потерями, использующие особенности графических данных. Продолжаются работы над алгоритмами сжатия с потерями, сохраняющими качество видео на высоком уровне.

Области применения методов кодирования и сжатия видеоинформации весьма разнообразны: от передачи и хранения видео до спутниковых цифровых телекоммуникационных систем. Внимание к сжатию видеоинформации особенно возросло в последнее десятилетие в связи с разработкой принципиально новых цифровых телекоммуникационных систем. Создание новейших цифровых устройств обработки, передачи и хранения видеоизображений связано с радикальным изменением технологических возможностей новейших процессорных систем, создаваемых ведущими мировыми фирмами, специализирующимися в области совершенствования аппаратных и программных компьютерных средств. Использование новейших процессоров с производительностью несколько миллиардов операций в се-

кунду обеспечивает реализацию самых сложных и вычислительно емких алгоритмов сжатия, что невозможно было осуществить ранее [4].

Передача цифрового видео от источника (видеокамера или записанный видеоролик) к получателю (видеодисплей) вовлекает в разработку целую цепь различных компонентов и процессов. Ключевыми звеньями этой цепи являются процесс компрессии (кодирования) и декомпрессии (декодирования), при которых несжатый цифровой видеосигнал сокращается до размеров, подходящих для его передачи и хранения, а затем восстанавливается для отображения на видеоэкране. Продуманная разработка процессов компрессии и декомпрессии может дать существенное коммерческое и техническое преимущество продукта, обеспечив лучшее качество видеоизображения, большую надежность и гибкую приспособляемость по сравнению с конкурирующими решениями. Таким образом, имеется заинтересованность в развитии и улучшении методов компрессии и декомпрессии видео [6].

1 Аналитический раздел

1.1 Цель разработки и основные решаемые задачи

Цель настоящей работы – разработать метод сжатия видео на основе кратномасштабного анализа. Задачи настоящей работы:

— провести анализ существующих методов сжатия видео,

— разработать метод сжатия видео на основе кратномасштабного анализа,

— создать приложение, позволяющее применять разработанный метод к видео,

— разработать модель для оценки предложенного метода,

— провести исследование работы метода.

1.2 Анализ алгоритмов сжатия видео

1.2.1 Характеристики видео

Видео (от лат. video — смотрю, вижу) — электронная технология формирования, записи, обработки, передачи, хранения и воспроизведения сигналов изображения.

Цифровое видео — множество технологий записи, обработки, передачи, хранения и воспроизведения визуального или аудиовизуального материала в цифровом представлении. Основное отличие от аналогового видео в том, что видеосигналы кодируются и передаются в виде последовательности бит. Цифровое видео может распространяться на различных видеоносителях, посредством цифровых видеоинтерфейсов в виде потока или файлов.

Этот тип данных имеет следующие характеристики:

— Разрешающая способность. Любой цифровой видеосигнал характеризуется разрешением (англ. resolution), горизонтальным и вертикальным, измеряемым в пикселях. Разрешение обозначается двумя числами, где первым числом обозначается количество точек в строке (горизонтальное разрешение), а вторым числом количество активных строк, участвующих в построении изображения (вертикальное разрешение).

— Количество цветов и цветовое разрешение. Количество цветов и цветовое разрешение видеозаписи описывается цветовыми моделями. Для различных стандартов применяются различные цветовые модели: YUV, YDbDr, YIQ, RGB (и ARGB), HSV, CMYK. Человеческий глаз может воспринять, по разным подсчётам, от 5 до 10 миллионов оттенков цветов. Количество цветов в видеозаписи определяется числом бит, отведённым для кодирования цвета каждого пикселя (англ. bits per pixel, bpp).

— Битрейт (ширина видеопотока или информационная скорость записи). Ширина видеопотока или битрейт (англ. bit rate) — это количество обрабатываемых бит видеоинформации за секунду времени (измеряется «бит/с» — бит в секунду или, чаще, «Мб/с» — мегабит в секунду; в английском обозначении bit/s и Mbit/s соответственно). Чем выше ширина видеопотока, тем, как правило, лучше качество видео. Различают два вида управления шириной потока в видеокодеке — постоянный битрейт (англ. constant bit rate, CBR) и переменный битрейт (англ. variable bit rate, VBR). Концепция VBR разработана для максимального сохранения качества видео, уменьшая при этом суммарный объём передаваемого видеопотока. При этом на быстрых сценах движения, ширина видеопотока возрастает, а на медленных сценах, где картинка меняется медленно, ширина потока падает. Это удобно для буферизованных видеотрансляций и передачи сохранённого видеоматериала по компьютерным сетям. Но для безбуферных систем реального времени и для прямого эфира (например, для телеконференций) это не подходит — в этих случаях необходимо использовать постоянную скорость видеопотока.

— Качество видео. Качество видеозаписи измеряется с помощью формальных метрик, таких как PSNR или SSIM, или с использованием субъективного сравнения с привлечением экспертов.

Качество видео может быть оценено субъективно по следующей методике:

а) выбираются видеопоследовательности для использования в тесте
б) выбираются параметры системы измерения
в) выбирается метод показа видео и подсчета результатов измерения
г) приглашается необходимое число экспертов (обычно не меньше 15)

д) проводится сам тест

е) подсчитывается средняя оценка на основе оценок экспертов.

1.2.2 Сжатие видео

Сжатие видео (англ. Video compression) — технология компрессии цифрового видео, позволяющая сократить количество данных, используемых для представления видеопотока. Сжатие видео позволяет эффективно уменьшать поток, необходимый для передачи видео по каналам связи, уменьшать пространство, необходимое для хранения данных на носителе.

Скорости передачи данных в сетях, а также емкости жестких дисков, флэш-памяти и оптических накопителей постоянно растут. Не смотря на снижение цены передачи и хранения бита информации, видеосжатие и его улучшение является необходимым. Видеосжатие имеет важные преимущества. Во-первых, оно дает возможность использовать цифровое видео в среде передачи и хранения видеоконтента, которая не поддерживает несжатое видео. Например, пропускная способность современного Интернета недостаточна для обращения с несжатым видео в реальном масштабе времени даже при низкой частоте кадра и малом его размере. Цифровой многослойный видеодиск DVD может вместить всего несколько секунд несжатого видео с разрешением и частотой кадров, обеспечивающими обычное телевизионное качество, поэтому использование DVD было бы абсолютно непрактичным без применения аудио и видеосжатия. Во-вторых, видеосжатие делает более эффективным использование ресурсов при передаче и хранении видеоданных. Если доступен высокоскоростной канал, то более привлекательным представляется решение, позволяющее передавать сжатое видео высокого разрешения вместо несжатого видео низкого разрешения. Несмотря на постоянный рост емкости устройств хранения информации и пропускной способности каналов передачи данных, представляется весьма вероятным, что сжатие видео остается существенным компонентом мультимедийных сервисов.

Сигнал, несущий определенную информацию, можно сжать путем удаления из него имеющейся избыточности. Избыточность – это компоненты данных, без которых можно обойтись для верного отображения исходной информации. Многие типы данных содержат в себе статистическую

избыточность. Такие данные могут быть эффективно сжаты с использованием алгоритмов сжатия без потерь. Сжатие без потерь применительно к видео не является эффективным. Поэтому для достижения высокой эффективности сжатия применяется сжатие с потерями. При сжатии видео с потерями используется несколько типов избыточности:

— когерентность областей изображения - малое изменение цвета изображения в соседних пикселях (свойство, которое используется в алгоритмах сжатия изображений с потерями),

— избыточность в цветовых плоскостях - используется важность яркости изображения для восприятия,

— подобие между кадрами - использование того факта, что на скорости 25 кадров в секунду, как правило, соседние кадры изменяются незначительно.

Использование подобия между кадрами в самом простом и наиболее часто используемом случае означает кодирование не самого нового кадра, а его разности с предыдущим кадром. Для видео, на котором нет большого количетсва движения (передача новостей, видеотелефоны, съемка с камер наблюдения) большая часть кадра остается неизменной, и такой метод позволяет значительно уменьшить поток данных. Более сложный метод заключается в нахождении для каждого блока в сжимаемом кадре наименее отличающегося от него блока в кадре, используемом в качестве базового. Далее кодируется разница между этими блоками. Этот метод существенно более ресурсоемкий.

При сравнении алгоритмов сжатия видео необходимо учитывать следующие особенности видео:

— Качество различных кадров одного и того же видео после обработки одним алгоритмом отличается. Это связано с несколькими факторами. Качество может меняться в зависимости от стратегии управления битрейтом. Пользователь выбирает разные стратегии битрейта, и в случае выбора постоянного битрейта на медленных сценах качество будет высоким, а на быстрых - низким. У кодеков есть так называемые ключевые кадры, качество которых обычно изменяется отдельно, и отличается от качества остальных кадров. На качество влияет префильтрация. Это означает, что

на любом достаточно длинном видео можно подобрать как достаточно хорошие, так и достаточно плохие кадры.

— Алгоритмы сжатия видео создаются для различных типов видео. Один и тот же алгоритм при различных параметрах, но одинаковом формате сжатия может по-разному сжимать различные видео, отличающиеся, например, наличием большого количества движения. Это означает, что для двух различных алгоритмов примерно равного качества можно подобрать видео, на которых первый будет заметно лучше второго и наоборот.

— Качество сжатия видео зависит от параметров кодирования. У алгоритмов есть достаточно много параметров, позволяющих при том же размере файла заметно изменить качество. Это параметры стратегии битрейта, префильтрации, управления частотой ключевых кадров, управление зависимости префильтра от видео и другие. Это означает, что для одного и того же видео и при использовании одного алгоритма при указании различных параметров размер файла может быть одинаковым, но сильно отличаться по качеству.

В настоящее время существуют следующие алгоритмы сжатия видео:

— сжатие без потерь

При декомпрессии результат в точности соответствует оригиналу. При сжатии без потерь невозможно получить высокие коэффициены сжатия. По этой причине практически всё широко используемое видео является сжатым с потерями. Краткое сравнение алгоритмов сжатия видео без потерь представлено в таблице 1.1.

а) HuffYUV

Каждый кадр сжимается отдельно при помощи алгоритма Хаффмана. Если часть кадра постоянно занимает однородный фон, то потребуется меньший поток данных, например для записей широкоэкранных кинофильмов (с чёрными полосами сверху и снизу изображения).

б) CorePNG

Каждый кадр видео сжимается с использованием алгоритма сжатия PNG, позволяя использовать все возможности PNG форма-

та, но также наследуя все его недостатки. CorePNG поддерживает кодирование видео в цветовом пространстве RGB с глубиной цвета 24 или 32 бит. 32-битная глубина позволяет использовать необязательный альфа-канал. Поддержка других цветовых пространств, таких как YUY2 или YV12, также предоставляется. CorePNG поддерживает запись P-кадров, называемых delta кадрами, в видеопотоке они кодируют только различия между предыдущим и текущим кадрами.

в) Lagarith

Базируется на Huffyuv, примерно сравним с ним по быстродействию, превосходит по степени сжатия. Хорошо сжимает видео с преобладанием статических изображений. Это достигается за счёт поддержки недействительных фреймов, то есть если предшествующий фрейм идентичен текущему, то он используется снова, а текущий отвергается. Lagarith работает в цветовых пространствах RGB24, RGB32, RGBA, YUY2 и YV12. Последние версии поддерживают многопроцессорность. Каждый кадр может быть отдельно декодирован, это облегчает поиск, вырезание, объединение.

г) Motion JPEG 2000

Это система кодирования видеоизображения с применением внутрикадровой технологии сжатия JPEG 2000, разработанная институтом интегральных схем общества Фраунгофера. В основе лежит дискретное вейвлет-преобразование, которое, в отличие от дискретно-косинусного преобразования, применяемого в кодеках семейства MPEG, производит кодирование сразу в двух областях: и в частотной, и в пространственной. Это исключает необходимость разбиения изображения на блоки. Для каждого кадра применяется только внутрикадровое сжатие без использования межкадрового кодирования. Отличительной особенностью является возможность масштабируемости, причем не только по размеру кадра, но и по качеству, то есть скорости видеопотока. Из-за особенностей вейвлет разложения, каждый кадр может содержать свои копии, уменьшенные вдвое по горизонтали и верти-

кали. Возможность сжатия без потерь позволяет применять данный кодек в области медицины, где требуется высокая точность изображений. Поддерживается вложенное аудио с синхронизацией с видео. Предлагаемые сферы применения: сохранение видеороликов в цифровых камерах, высококачественная видеозапись и покадровое редактирование, цифровой кинематограф, медицинские и спутниковые изображения

Таблица 1.1 — Сравнение алгоритмов сжатия видео без потерь

Алгоритм	Сжатие	Достоинства	Недостатки
HuffYUV	каждый кадр, алгоритм Хаффмана	сжатие видео с наличием областей одного цвета на кадрах	не использует избыточность информации между кадрами
CorePNG	каждый кадр, алгоритм PNG	наличие прозрачности, возможность гамма коррекции	не использует избыточность информации между кадрами
Lagarith	основан на Huffyuv, использует предыдущий кадр при отсутствии изменений	хорошее сжатие при отсутствии движения	не использует избыточность информации между кадрами
Motion JPEG 2000	каждый кадр, алгоритм JPEG 2000	задается степень сжатия, нет дробления изображения на блоки	сложность алгоритма, не использует избыточность информации между кадрами

— сжатие с потерями

Одна из наиболее мощных технологий, позволяющая повысить степень сжатия, — это компенсация движения. При любой современной системе сжатия видео последующие кадры в потоке используют похожесть областей в предыдущих кадрах для увеличения степени сжатия. Однако, из-за движения каких-либо объектов в кадре (или самой камеры) использование подобия соседних кадров было неполным. Технология компенсации движения позволяет находить похожие участки, даже если они сдвинуты относительно предыдущего кадра. Краткое сравнение алгоритмов сжатия видео с потерями представлено в таблице 1.2.

а) M-JPEG

Каждый кадр сжимается отдельно с помощью алгоритма сжатия изображений JPEG. Основным преимуществом видеосжатия Motion JPEG является простота реализации, что делает MJPEG

подходящим для реализации в устройствах с ограниченными вычислительными ресурсами. Однако при отсутствии межкадрового сжатия достижение заданного битрейта требует использования большего, чем в случае MPEG, покадрового сжатия, что приводит появлению заметных артефактов сжатия. Недостатками MJPEG являются более низкий коэффициент сжатия по сравнению с потоковыми методами сжатия, например, MPEG-4 и проявляющиеся при высоких степенях сжатия артефакты.

б) MPEG-1

Группа стандартов цифрового сжатия аудио и видео. Применяется для кодирования с низким разрешением и низким битрейтом, может использоваться при любом разрешении. Поддерживает только прогрессивную развертку.

Перед началом кодирования происходит анализ видеоинформации, выбираются ключевые кадры, которые не будут изменяться при сжатии, а так же кадры, при кодировании которых часть информации будет удаляться. Всего выделяется три типа кадров:

Определяет формат видеопотока, который может быть представлен как три типа кадра — независимо сжатые кадры (I-кадры), кадры, сжатые с использованием предсказания движения в одном направлении (P-кадры) и кадры, сжатые с использованием предсказания движения в двух направлениях (B-кадры). Соответствующие группы кадров от одного I-кадра до другого образуют GOP — Group Of Pictures — группу кадров.

в) MPEG-2

Основан на MPEG-1. MPEG-2 поддерживает видео и в прогрессивной, и в чересстрочной развёртке. Требует значительные ресурсы для декомпрессии. Плохое качество при малом разрешении видео.

г) MPEG-4

Разрабатывался для передачи потокового видео по каналам с низкой пропускной способностью. При одном и том же битрейте и определённых условиях кодирования, качество изображения фильма в MPEG-4 может быть сравнимо или даже лучше, чем в

случае применения MPEG-1 или MPEG-2. Алгоритм компрессии видео аналогичен MPEG-1 и MPEG-2. При кодировании исходного изображения сохраняет ключевые кадры, а вместо сохранения промежуточных — прогнозирует и сохраняет лишь информацию об изменениях в текущем кадре по отношению к предыдущему. Может возникать ступенчатость при медленном цветовом переходе, появляться артефакты при наличии ошибочных кадров последовательности.

д) DV

Используется для сжатия в цифровых камерах. Технически схож с MJPEG, но не совместим с ним. Формат DV содержит дополнительные возможности: наложение титров, специальный способ синхронизации аудио и видео информации. В DV используются фиксированные параметры: 720x480 при 29.97 FPS (или 720x576 при 25 FPS) при скорости 25MBit/second.

Таблица 1.2 — Сравнение алгоритмов сжатия видео с потерями

Алгоритм	Сжатие	Достоинства	Недостатки
M-JPEG	каждый кадр, алгоритм JPEG	простота реализации	не использует избыточность информации между кадрами, артефакты сжатия
MPEG-1	разбивает кадры по типам, сжимает группы кадров	может использоваться при любом разрешении	поддерживает только прогрессивную развертку
MPEG-2	схож с MPEG-1	поддерживает видео и в прогрессивной, и в чересстрочной развёртке	требует значительные ресурсы для декомпрессии, плохое качество при малом разрешении видео
MPEG-4	схож с MPEG-1	улучшен по сравнению с MPEG-1 и MPEG-2	ступенчатость при медленном цветовом переходе, артефакты при наличии ошибочных кадров
DV	схож с MJPEG	наложение титров, специальный способ синхронизации аудио и видео информации	специализирован для цифровых камер

Для различных видео один и тот же метод может давать как высокую степень сжатия, так и увеличивать размер исходных данных. Для оценки работы метода необходимо разделить видео на классы. Так как гра-

фические данные разнообразны, дадим следующие параметры для классификации:

— битрейт (частота кадров, разрешение видео),
— количество изменений между кадрами (движение),
— тип видео (реальная съемка, мультипликация).

1.3 Виды вейвлет-преобразований

1.3.1 Непрерывное вейвлет-преобразование

Непрерывное вейвлет-преобразование (CWT) есть скалярное произведение сигнала $f(x)$ и базисных функций, указанных в формуле 1.1.

$$\psi_{a,b}(x) = a^{-1/2}\psi\Big(\frac{x-b}{a}\Big), a \in R^{+}, b \in R \tag{1.1}$$

Таким образом, непрерывное вейвлет-преобразование может быть расчитано по формуле 1.2.

$$CWT_f(a,b) = a^{-1/2}\int_{-\infty}^{+\infty}\psi\Big(\frac{x-b}{a}\Big)f(x)d(x) \tag{1.2}$$

Базисные функции $\psi_{a,b}$ являются вещественными и колеблются вокруг оси абсцисс. Они определены на некотором интервале. Данные функции называются вейвлетами (короткие волны) и могут рассматриваться как масштабированные и сдвинутые версии функции прототипа $\psi(x)$. Параметр b показывает расположение во времени, а a - параметр масштаба. Большие значения a соответствуют низким частотам, малые – высоким.

Для того чтобы было возможно обратное получение $f(x)$ из результата CWT, функция $\psi(x)$ должна удовлетворять условию 1.3, где через $\Psi(\omega)$ обозначено преобразование Фурье $\psi(x)$.

$$C_\psi = \int_0^\infty \frac{|\Psi(\omega)|^2}{\omega}d\omega < \infty \tag{1.3}$$

Если $\psi(x)$ – локальная функция, то из 1.3 следует, что ее среднее значение равно нулю: $\int_{-\infty}^{\infty}\psi(x)dx = 0$. Тогда формула реконструкции име-

ет вид 1.4

$$f(x) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{0}^{\infty} CWT_f(a,b) a^{-1/2} \psi\Big(\frac{x-b}{a}\Big) \frac{dadb}{a^2} \tag{1.4}$$

В большинстве приложений сигнал, в том числе изображение, является дискретным. Поэтому вместо непрерывного вейвлет-преобразования используется дискретное вейвлет-преобразование [7].

1.3.2 Дискретное вейвлет-преобразование

Под кратномасштабным анализом понимается описание пространства $L^2(R)$ через иерархически вложенные подпространства V_m, которые не пересекаются и дают в пределе $L^2(R)$, то есть $\cap_{m \in Z} V_m = \{0\}, \overline{\cup_{m \in Z}} = L^2(R)$. Эти пространства имеют следующие свойства. Для любой функции $f(x) \in V_m$ ее сжатая версия будет принадлежать пространству V_{m-1}, $f(x) \in V_m \Leftrightarrow f(2x) \in V_{m-1}$. Существует такая функция $\phi(x) \in V_0$, что ее сдвиги $\phi_{0,n}(x) = \phi(x-n), n \in Z$ образуют ортонормированный базис пространства V_0. Так как функции $\phi_{0,n}(x)$ образуют ортонормированный базис пространства V_0, то функции $\phi_{m,n}(x) = 2^{-m/2}\phi(2^{-m}x - n)$ образуют ортонормированный базис пространства V_m. Эти базисные функции называются масштабирующими, так как они создают масштабированные версии функций в $L^2(R)$. Пусть имеется некоторая непрерывная функция $f_0(x)$. Дискретный сигнал c_n может быть представлен как последовательность коэффициентов при масштабирующих функциях, по которым раскладывается $f_0(x)$: $f_0(x) = \sum_n c_{0,n}\phi_{0,n}(x)$, где $c_{0,n} = c_n$. Сигнал интерпретируется как последовательность коэффициентов разложения, полученная в ходе кратномасштабного анализа функции $f_0(x)$. Функция $f_0(x)$ декомпозируется: $f_0(x) = f_1(x) + e_1(x) = \sum_k c_{1,k}\phi_{1,k}(x) + \sum_k d_{1,k}\psi_{1,k}(x)$. Таким образом, получили две новые последовательности $c_{1,n}$ и $d_{1,n}$. Этот процесс может быть продолжен по $f_1(x)$. Функция $f_0(x)$ будет представлена совокупностью коэффициентов $d_{m,n}, m \in Z^+, n \in Z$. Вычисление коэффициентов $c_{j,k}$ и $d_{j,k}$ и возможно итеративно без использования функций $\phi(x)$ и $\psi(x)$. Для произвольного j:

$$c_{j,k} = 2_{1/2} \sum_n cj - 1,nh_{n+2k} \tag{1.5}$$

$$d_{j,k} = 2_{1/2} \sum_n cj - 1,ng_{n+2k}. \tag{1.6}$$

Таким образом, процесс декомпозиции полностью дискретный. Последовательности h_n и g_n называются фильтрами. На фильтры налагаются ограничения: $2\sum_n(h_{n+2k}h_{p+2k} + g_{n+2k}g_{p+2k}) = \delta_{n,p}$, $2\sum_n h_{n+2k}h_{n+2p} = 2\sum_n g_{n+2k}g_{n+2p} = \delta_{k,p}$, $2\sum_n h_{n+2k}h_{n+2p} = 0$ [7].

Цель сжатия - выразить исходный набор данных через меньший набор данных с потерей информации либо без потери. Имеем функцию $f(x)$, выраженную через взвешенную сумму базисных функций $u_1(x)$,...,$u_m(x)$: $f(x) = \sum_{i=1}^m c_i u_i(x)$. Набор данных в этом случае состоит из коэффициентов $c1$,...,c_m. Необходимо найти функцию, аппроксимирующую $f(x)$, с меньшим числом коэффициентов. При данной, установливаемой пользователем, допустимой погрешности ε (в случае сжатия без потерь $\varepsilon = 0$) требуется функция $\hat{f}(x) = \sum_{i=1}^{\hat{m}} \hat{c}_i \hat{u}_i(x)$ такая, что $\hat{m} < m$ и $\|f(x) - \hat{f}(x)\| \leq \varepsilon$ в некоторой норме. Под сжатием понимается задача уменьшения числа коэффициентов, необходимых для функции. Необходимо упорядочить коэффициенты c_1,...,c_m так, чтобы для каждого $\hat{m} < m$ первые $\hat{m}$ элементов последовательности давали наилучшее приближение $\hat{f}(x)$ к $f(x)$ в L^2-норме. Решение этой задачи является прямым, если базис ортонормированный.

Пусть $\pi(i)$ - это перестановка 1,...,m, а $\hat{f}(x)$ - функция, использующая коэффициенты, соответствующие первым $\hat{m}$ членам перестановки $\pi(i)$, тогда $\hat{f}(x) = \sum_{i=1}^{\hat{m}} c_{\pi(i)} u_{\pi(i)}$.

Квадрат L^2-погрешности в этом приближении определяется как

$$\begin{aligned} \|f(x) - \hat{f}(x)\|_2^2 = \langle f(x) - \hat{f}(x) | f(x) - \hat{f}(x) \rangle = \\ \langle \sum_{i=\hat{m}+1}^{m} c_{\pi(i)} u_{\pi(i)} | \sum_{j=\hat{m}+1}^{m} c_{\pi(j)} u_{\pi(j)} \rangle = \\ \sum_{i=\hat{m}+1}^{m} \sum_{j=\hat{m}+1}^{m} c_{\pi(i)} c_{\pi(j)} \langle u_{\pi(i)} | u_{\pi(j)} \rangle = \\ \sum_{i=\hat{m}+1}^{m} \left(c_{\pi(i)}\right)^2 \end{aligned} \tag{1.7}$$

Последний шаг обуславливается допущением, что базис является ортонормированным, значит $\langle u_i | u_j \rangle = \delta_{ij}$. Таким образом, квадрат общей L^2-ошибки равен сумме квадратов всех тех коэффициентов, которые будут выбраны для децимации. Для того, чтобы минимизировать погрешность для любого данного $\hat{m}$, необходимо выбрать такую перестановку $\pi(i)$, которая располагает коэффициенты в порядке уменьшения их величины: то есть $\pi(i)$ удовлетворяет $|c_{\pi l}| \geq ... \geq |c_{\pi m}|$. На рисунке 1.1 показано, как можно преобразовать одномерную функцию в коэффициенты с использованием функции нормированного базиса Хаара. L^2 сжатие к полученным коэффициентам применяется простым удалением или игнорированием коэффициентов с самыми маленькими величинами. Варьирование степени сжатия дает последовательность приближений к исходной функции [8]. Расчитанные коэффициенты для вейвлетов Добеши представлены в приложении А. Для вейвлетного преобразования была выбрана система из четырех коэффициентов [9].

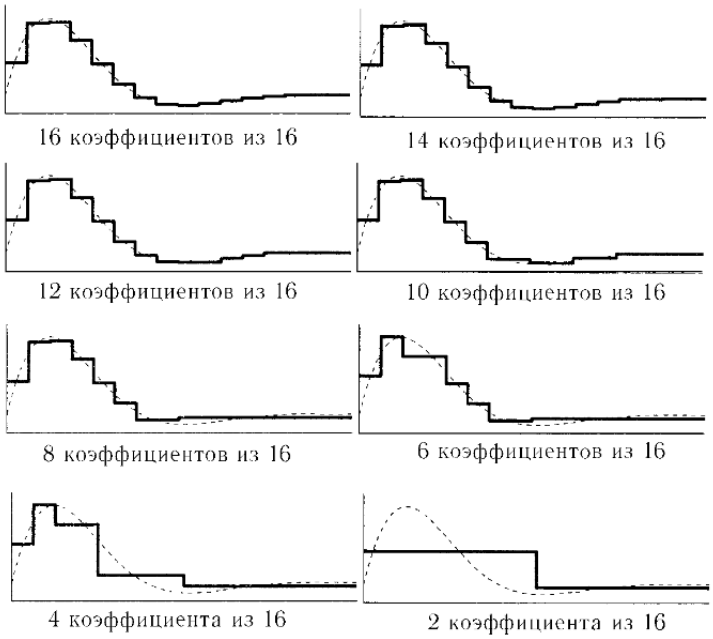

Рисунок 1.1 — Преобразование одномерной функции в коэффициенты с использованием функции нормированного базиса Хаара

1.3.3 Матричное представление DWT

Пусть v_j – последовательность конечной длины $c_{j,n}$ для некоторого J. Этот вектор преобразуется в вектор v_{j+1}, содержащий последовательности $c_{j+1,n}$ и $d_{j,n}$, каждая из которой половинной длины. Преобразование мо-

жет быть записано в виде матричного умножения $v_{j+1} = M_j v^j$, где матрица M_j – квадратная и состоит из нулей и элементов h_n, является ортонормированной, и обратная ей матрица является транспонированной. В формулах 1.8 и 1.9 представлен пример прямого и обратного преобразования для фильтра длиной $L = 4$, последовательности длиной $N = 8$, начального значения $j = 0$.

$$\begin{bmatrix} c_{10} \\ c_{11} \\ c_{12} \\ c_{13} \\ d_{10} \\ d_{11} \\ d_{12} \\ d_{13} \end{bmatrix} = \sqrt{2} \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & & & & \\ & & h_0 & h_1 & h_2 & h_3 & & \\ & & & & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & & & & & h_0 & h_1 \\ h_3 & -h_2 & h_1 & -h_0 & & & & \\ & & h_3 & -h_2 & h_1 & -h_0 & & \\ & & & & h_3 & -h_2 & h_1 & -h_0 \\ h_1 & h_0 & & & & & h_3 & -h_2 \end{bmatrix} \begin{bmatrix} c_{00} \\ c_{01} \\ c_{02} \\ c_{03} \\ c_{04} \\ c_{05} \\ c_{06} \\ c_{07} \end{bmatrix} \tag{1.8}$$

Выражения 1.5 и 1.6 – это один шаг DWT. Полное DWT заключается в итеративном умножении верхней половины вектора v^{j+1} на квадратную матрицу M_{j+1}, размер которой 2^{d-j}. Эта процедура может повторяться d раз, пока длина вектора не станет равна единице [7].

$$\begin{bmatrix} c_{00} \\ c_{01} \\ c_{02} \\ c_{03} \\ c_{04} \\ c_{05} \\ c_{06} \\ c_{07} \end{bmatrix} = \sqrt{2} \begin{bmatrix} h_0 & & & h_2 & h_3 & & & h_1 \\ h_1 & & & h_3 & -h_2 & & & -h_0 \\ h_2 & h_0 & & & h_1 & h_3 & & \\ h_3 & h_1 & & & -h_0 & -h_2 & & \\ & h_2 & h_0 & & & h_1 & h_3 & \\ & h_3 & h_1 & & & -h_0 & -h_2 & \\ & & h_2 & h_0 & & & h_1 & h_3 \\ & & h_3 & h_1 & & & -h_0 & -h_2 \end{bmatrix} \begin{bmatrix} c_{10} \\ c_{11} \\ c_{12} \\ c_{13} \\ d_{10} \\ d_{11} \\ d_{12} \\ d_{13} \end{bmatrix} \tag{1.9}$$

1.3.4 Построение фильтров Добеши

В настоящее время разработаны различные вейвлет-функции, ориентированные для решения различных задач. В данной работе для рассмотрения выбраны вейвлеты Добеши, так как предполагается, что увеличение

числа ненулевых моментов для Psi приводит к лучшей сжимаемости [1]. Добеши были сконструированы ортонормированные вейвлеты с компактными носителями на бесконечной вещественной оси.

Для построения фильтров Добеши процесс декомпозиции может быть записан как $v^{j+1} = M_j v^j$. Из свойств M:

$$M_j^T v^j M_j = I \tag{1.10}$$

I - единичная матрица. Рассмотрим фильтр длиной 4. Получим матрицу, аналогичную 1.8. Из равенства 1.10 следует, что $h_0^2+h_1^2+h_2^2+h_3^2 = 1/2$, $h_0h_1 + h_2h_3 = 1/2$. Эта система не дает единственного решения для фильтра h_m. Если потребовать, чтобы высокочастотный фильтр имел два нулевых момента: $h_3 - h_2 + h_1 - h_0 = 0$, $h_3 - 2h_2 + 3h_1 - 4h_0 = 0$, тогда: $h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$, $h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$, $h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$, $h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$. Эти коэффициенты определяют вейвлеты, называемые D4-вейвлетами Добеши. Все фильтры Добеши могут быть получены посредством этой процедуры. Для фильтра длиной $L = 2N$ необходимо N равенств относительно нулевых моментов для получения единственного решения. В общем случае решение может быть получено численно.

В D_4-конструкции задействованы следующие последовательности:

$$\begin{aligned} p = a &= \frac{1}{4\sqrt{2}}(1+\sqrt{3}, 3+\sqrt{3}, 3-\sqrt{3}, 1-\sqrt{3}), \\ q = b &= \frac{1}{4\sqrt{2}}(1-\sqrt{3}, -3+\sqrt{3}, 3+\sqrt{3}, -1-\sqrt{3}) \end{aligned} \tag{1.11}$$

Последовательность p представляет ненулевые элементы каждого столбца P^j, тогда как последовательность q содержит ненулевые элементы столбца Q^j. Аналогично, последовательности a и b представляют строки A^j и B^j соответственно. Последовательность для вейвлет фильтра можно получить, если поменять порядок элементов в последовательности масштабирующих функций и изменить их знаки на противоположные. Такие последовательности носят название квадратурных зеркальных фильтров.

D_4-масштабирующая функция и D_4-вейвлет изображены на рисунке 1.2. Обе функции ассиметричны и всюду недифференцируемы, хотя при этом они являются непрерывными. Эти базисные функции являются члена-

ми целого семейства базисов, полученных Добеши, включающего базисы, гладкость которых возрастает с увеличением носителей базисных функций.

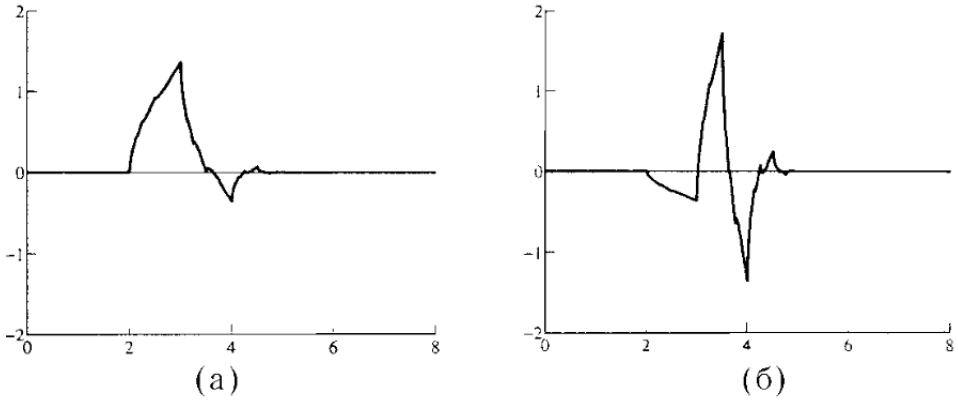

Рисунок 1.2 — Функции базиса вейвлетов Добеши: (а) D_4-масштабирующая функция; (б) D_4-вейвлет

1.3.5 Представление DWT для обработки двумерных и трехмерных данных

Исходное изображение, например, один кадр из видео, может быть рассмотрено как одномерный массив, к которому затем применяется вейвлетное преобразование. В этом случае происходит перераспределение данных изображения, и не используется особенность избыточности данных изображений в двух измерениях. При применении вейвлетного преобразования к изображениям будем поочередно преобразовывать строки и столбцы исходного изображения, затем уменьшать рассматриваемую область в два раза. В начале рассматриваемая область соответствует всему изображению. На рисунке 1.3 представлена схема применения к изображению последовательности низкочастотных и высокочастотных фильтров. Обозначим высокочастотный фильтр H, а низкочастотный фильтр L. Тогда LH_x – вертикальное высокочастотное фильтрование, к которому применяется горизонтальное низкочастотное фильтрование, HL_x – вертикальное низкочастотное фильтрование, к которому применяется горизонтальное высокочастотное фильтрование [5]. При обратном вейвлет-преобразовании к данным применяется та же последовательность действий в обратном порядке.

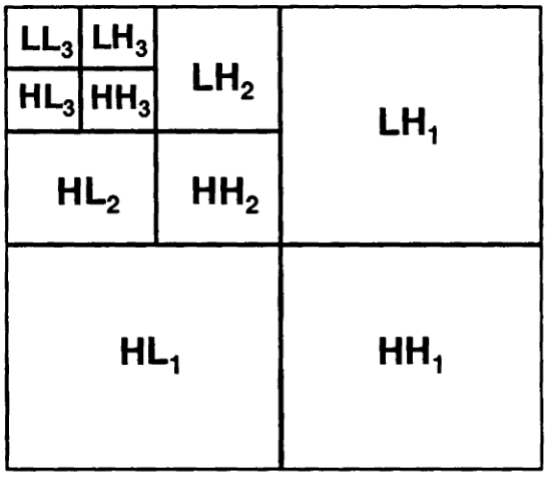

Рисунок 1.3 — Схема применения к изображению последовательности низкочастотных L и высокочастотных H фильтров. На схеме LH_x – вертикальное высокочастотное фильтрование, к которому применяется горизонтальное низкочастотное фильтрование, HL_x – вертикальное низкочастотное фильтрование, к которому применяется горизонтальное высокочастотное фильтрование

Для использования кратномасштабного анализа при обработке видео необходимо реализовать 3D-версию анализа и синтеза банков фильтров. В случае трехмерного преобразования вейвлет преобразование применяется к каждому из трех измерений. Если данные имеют размеры $N_1 \times N_2 \times N_3$, то применение вейвлет преобразования к первому измерению дает два набора данных, каждый из которых имеет размер $N_1/2 \times N_2 \times N_3$. Применение вейвлет преобразования ко второму измерению дает четыре набора данных, каждый из которых имеет размер $N_1/2 \times N_2/2 \times N_3$. Применение вейвлет преобразования к третьему измерению дает восемь наборов данных, каждый из которых имеет размер $N_1/2 \times N_2/2 \times N_3/2$. Это показано на рисунке 1.4, где X - исходный набор данных, W - набор данных после применения вейвлет преобразования, индекс L - набор данных после применения низкочастотного фильтра, индекс H - набор данных после применения высокочастотного фильтра.

1.4 Достоинства и недостатки вейвлет-преобразований

— Вейвлет-преобразования обладают всеми достоинствами преобразования Фурье

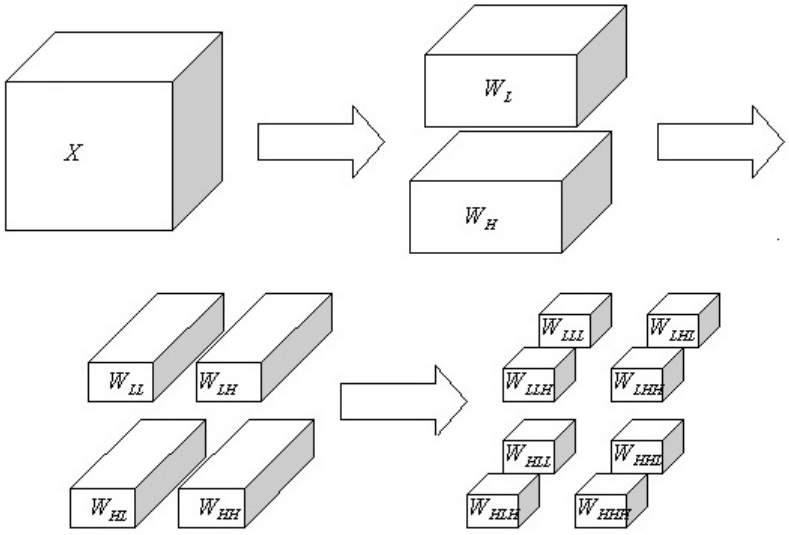

Рисунок 1.4 — Схема применения к трехмерным данным последовательности низкочастотных L и высокочастотных H фильтров, где X - исходный набор данных, W - набор данных после применения вейвлет преобразования, индекс L - набор данных после применения низкочастотного фильтра, индекс H - набор данных после применения высокочастотного фильтра

— Вейвлетные базисы могут быть хорошо локализованными как по частоте, так и по времени. При выделении в сигналах разномасштабных процессов можно рассматривать выбранные масштабные уровни разложения

— Вейвлетные базисы имеют разнообразные базовые функции, свойства которых ориентированы на решение различных задач

— Недостатком вейвлет-преобразований является их относительная сложность

1.5 Метрики оценки качества видео

Наиболее часто используемые сравнительные оценки качества двух изображений – это среднеквадратическая ошибка и пиковое отношение сигнал-шум. Методы, основанные на этих оценках, хороши для изображений, имеющих белый шум. Однако эти меры некорректно отражают структурные искажения при кодировании (сжатии), а также плохо коррелируют с визуальной оценкой качества.

Важнейшими характеристиками при сравнении различных методов кодирования видео сигналов является выходной битрейт и оценки качества восстановленного после декодирования видео потока. Одним из про-

стых критерием оценки потери качества является среднеквадратическое отклонение значений пикселей сжатого изображения от оригинала. По этому критерию изображение будет сильно испорчено при изменении яркости всего на 5%. В тоже время изображение со снегом, резким изменением цвета отдельных точек будут признаны почти не изменившимися. Другим критерием является максимальное отклонение от оригинала. Даная мера крайне чувствительна к биению отдельных пикселей, т.е. в изображении может измениться только один пиксель, и данный критерий признает изображение сильно испорченным. На практике используемой мерой качества изображения является критерий соотношения сигнал/шум (PSNR). Эта мера аналогична среднеквадратическому отклонению, но пользоваться ей удобнее из-за логарифмического масштаба шкалы. Лучше всего потери в качестве оценивает человеческий глаз. Сжатие изображение можно считать отличной, если на глаз невозможно отличить оригинал от сжатого изображения. Но на практике при сжатии с потерями в изображение всегда вносятся какие-либо искажения заметные при сравнении оригинала и сжатого изображения. Среди большого числа критериев оценки качества восстановленного изображения наибольшее распространение получили среднеквадратическое отклонение и соотношение сигнал/шум, а для визуальной оценки используется разностное изображение. На практике, видео изображение со значениями PSNR порядка 40-43 дБ и выше является изображением очень высокого качества. Сжатое изображение с уровнем PSNR свыше 43 дБ в связи с особенностями человеческого зрения неотличимо от оригинала, с уровнем от 40 до 43 дБ показывает отличное качество изображения, с уровнем PSNR от 35 до 40 дБ – хорошее, от 30 до 35 дБ - приемлемое качество изображения, с уровнем PSNR ниже 30 дБ – неприемлемое качество изображения [2].

1.5.1 PSNR

Метрика, которую часто используют на практике, называется мерой отношения сигнала к шуму (peak-to-peak signal-to-noise ratio — PSNR).

$$PSNR = 10 \log_{10} \frac{MaxErr^2 wh}{\sum_{i=0,j=0}^{w,h} (x_{i,j} - y_{i,j})^2} \tag{1.12}$$

PSNR рассчитывается по формуле 1.12 где MaxErr - максимум модуля разности цветовой компоненты, w - ширина видео, h - высота видео. Данная метрика, по сути, аналогична среднеквадратичному отклонению, однако пользоваться ей несколько удобнее за счет логарифмического масштаба шкалы. Ей присущи те же недостатки, что и среднеквадратичному отклонению.

1.5.2 MSAD

Значением данной метрики является усреднённая абсолютная разность значений цветовых компонент в соответствующих точках сравниваемых изображений. Используется, например, для отладки кодеков или фильтров. Расчитывается по формуле 1.13

$$d(X,Y) = \frac{\sum_{i=1,j=1}^{m,n} \left| X_{i,j} - Y_{i,j} \right|}{mn} \tag{1.13}$$

1.5.3 SSIM INDEX

Индекс структурного сходства (от англ. SSIM - structure similarity) является одним из методов измерения схожести между двумя изображениями. SSIM индекс это метод полного сопоставления, другими словами, он проводит измерение качества на основе исходного изображения (не сжатого или без искажений). SSIM индекс является развитием традиционных методов, таких как PSNR и метод среднеквадратичной ошибки MSE, которые оказались несовместимы с физиологией человеческого восприятия. Отличительной особенностью метода, помимо упомянутых ранее (MSE и PSNR), является то, что метод учитывает "восприятие ошибки благодаря учёту структурного изменения информации. Идея заключается в том, что пиксели имеют сильную взаимосвязь, особенно когда они близки пространственно. Данные зависимости несут важную информацию о структуре объектов и о сцене в целом. SSIM метрика рассчитана на различные размеры окна. Разница между двумя окнами x и y имеющими одинаковый размер $N \times N$. Расчитывается по формулам 1.14

$$SSIM = \left(\frac{2\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2}\right)\left(\frac{2\bar{X}\bar{Y}}{\left(\bar{X}\right)^2+\left(\bar{Y}\right)^2}\right)\left(\frac{\sigma_{xy}}{\sigma_x\sigma_y}\right); \tag{1.14}$$

$$\begin{aligned}
\bar{X} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij}, \bar{Y} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} y_{ij};\\
\sigma_x^2 = \frac{1}{(M-1)(N-1)}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{ij}-\bar{X})^2;\\
\sigma_y^2 = \frac{1}{(M-1)(N-1)}\sum_{i=1}^{M}\sum_{j=1}^{N}(y_{ij}-\bar{Y})^2;\\
\sigma_{xy} = \frac{1}{(M-1)(N-1)}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{ij}-\bar{X})(y_{ij}-\bar{Y}),
\end{aligned} \tag{1.15}$$

где SSIM – значение меры сходства (качества) изображений; $X = \{x_{ij}\}$ и $Y = \{y_{ij}\}$ – сравниваемые изображения; M, N – размеры изображения. Первая составляющая выражения 1.14 является коэффициентом корреляции между изображениями X и Y. Вторая составляющая характеризует сходство средних значений яркостей двух сравниваемых изображений. Третья составляющая характеризует сходство контрастов двух сравниваемых изображений.

Чем выше значение меры сходства изображений, тем лучше выполнена обработка изображения. В общем случае мера сходства изображений рассчитывается в непересекающихся областях для каждого изображения отдельно. Поскольку на изоб- ражениях сцены объекты не изменялись и не двигались, меру можно вычислять сразу на всем изображении.

1.6 Метод сжатия видео на основе кратномасштабного анализа

В разрабатываемой системе необходимо выделить параметры, которые влияют на поведение системы. Были выделены следующие параметры:

— Размер блока
— Порядок системы коэффициентов для вейвлетного преобразования
— Процент сохраняемых данных
— Алгоритм кодирования без потерь

Эти параметры передаются как входные данные вместе с графическими данными.

Вейвлетное преобразование является эффективным, если имеется избыточность графических данных [10]. Для трехмерного вейвлетного преобразования, в частности, преобразования блоков в видео, необходима избыточность в трех измерениях. Это означает что соседние пиксели на одном кадре и на смежных кадрах должны иметь одинаковое или близкое значение цветовых компонент.

Для определения избыточности на видео необходимо провести предварительный анализ графических данных. На видео необходимо найти такие блоки, где избыточность во всех трех измерениях будет максимальной. Такие блоки могут быть выделены следующим образом. На видео необходимо найти смены сцен и планов. План - это система условного деления кинематографического пространства (то есть пространства, представленного на экране). Предполагается, что избыточность в данных будет присутствовать только пока последовательность кадров представляет один и тот же план. Как только происходит смена плана, в данных между кадрами наблюдается резкая граница, которая требует дополнительной информации при кодировании. Таким образом, во время анализа данных необходимо найти смены планов. Каждая смена означает границу определенного блока.

Для поиска границ между блоками необходимо определить способ сравнения кадров. Для этой задачи могут быть использованы метрики оценки качества видео, описанные ранее, такие как PSNR, MSAD, SSIM INDEX.

Необходимо также определить минимальный размер блока и максимальный размер блока. Минимальный размерблока требуется для устранения большого количество блоков малого размера, что посребует вычислительных ресурсов для их обработки. Максимальный размер блока определяется мощностью машины, на которой производятся вычисления.

Для поиска границ между блоками будем последовательно просматривать кадры видео и вычислять разницу между соседними кадрами по одной из метрик. Определим некоторое пороговое значение σ, превышение этого значения при вычислении разницы между кадрами будет озна-

чать наличие границы между блоками. Общая схема поиска блоков на видео представлена на рисунке 1.6.

Преобразование видео с использованием вейвлетных преобразований состоит из двух основных этапов – прямого преобразования и обратного преобразования, общая схема представлена на рисунке 1.5. Последовательное применение прямого и обратного преобразований не изменяет данные. Прямое преобразование с последующим квантованием и децимацией позволяет сжать исходные данные. Качество видео при этом зависит от параметров кодирования.

Рисунок 1.5 — Общая схема компрессии и декомпрессии видео

Вейвлетное преобразование для данных в третьем измерении, то есть для видео по времени, требует разбиения видео на блоки по кадрам. Это необходимо, так как видео, как правило, содержит большое число кадров. Это означает, что вейвлетное преобразование видео по времени потребует большое количесво вычислительных ресурсов. После разбиения на блоки необходимо применить вейвлет преобразование к каждому блоку в трех измерениях.

После применения вейвлет преобразования к блоку будет получен блок данных, содержащий информацию о высоких и низких частотах в бло-

ке. Применение обратного преобразования на этом этапе позволит получить исходный набор данных.

Далее необходимо провести децимацию данных в преобразованном блоке. Перед началом работы пользователем определяется процент данных, который будет удален из каждого блока. Для проведения децимации необходимо найти все минимальные коэффициенты и обнулить их. Их количество должно соответствовать указанному проценту удаляемых данных. Блок данных после децимации содержит нули и может быть закодирован. При достаточно большом количестве нулевых данных могут использоваться специальные структуры данных для хранения разреженных данных. Далее необходимо сжать получившиеся данные с помощью алгоритмов сжатия без потерь. Это позволяет максимально уменьшаеть объем памяти, занимаемый итоговым файлом на диске, однако уменьшает скорость декомпрессии.

Для воспроизведения видео необходимо провести действия в обратном порядке. Сначала необходимо провести декомпрессию преобразованных данных выбранным алгоритмом сжатия без потерь. Затем к каждому преобразованному блоку необходимо применить обратное вейвлетное преобразование. После чего блоки должны быть объединены в общую последовательность кадров.

Рисунок 1.6 — Схема поиска блоков на видео

2 Конструкторский раздел

2.1 Требования к разрабатываемому программному обеспечению

К разрабатываемому программному обеспечению предъявляются следующие высокоуровневые требования:

— разрабатываемое программное обеспечение должно обеспечивать функционирование по требованию пользователя,

— программное обеспечение должно производить кодирование и декодирование указанного файла с графическими данными,

— программное обеспечение должно проводить анализ графических данных и осуществлять разбиение графических данных на блоки,

— программное обеспечение должно выполнять прямое и обратное вейвлетное преобразование блоков графических данных,

— программное обеспечение должно выполнять децимацию графических данных после применения вейвлетного преобразования.

2.2 Пользовательские требования к программному обеспечению

Пользовательские требования (функциональные требования с точки зрения пользователей) описывают цели и задачи пользователей программного обеспечения, которые должны достигаться и выполняться пользователями при помощи разрабатываемого программного обеспечения. К разрабатываемому программному обеспечению предъявляются следующие требования:

— разрабатываемое программное обеспечение должно позволять пользователю выбирать файл с графическими данными,

— программное обеспечение должно позволять пользователю задавать процент сохраняемых данных,

— программное обеспечение должно позволять пользователю выполнять кодирование файла с графическими данными с помощью разработанного метода с использованием кратномасштабного анализа,

— программное обеспечение должно позволять пользователю выполнять декодирование файла с преобразованными графическими данными,

— программное обеспечение должно выполнить воспроизведение графических данных после декодирования.

2.3 Входные и выходные параметры

Разрабатываемое программное обеспечение состоит из двух частей:

— кодирование графических данных,

— декодирование графических данных.

Входными данными для функционала кодирования графических данных являются исходный файл с графическими даннными и процент сохраняемых данных. Выходными данными для функционала кодирования графических данных является файл, содержащий кодированную последовательность графических данных после применения вейвлетного преобразования и децимации. Входными данными для функционала декодирования графических данных являются исходный файл с кодированными графическими даннными. Выходными данными для функционала декодирования графических данных является последовательность кадров, состоящая из графических данных, полученных после применения обратного вейвлетного преобразования.

2.4 Сценарии функционирования

Сценарии функционирования разрабатываемого программного комплекса представлены на рисунке 2.1.

Возможны следующие сценарии использования программного обеспечения:

— Прецедент 1 - Кодировать данные

Прецедент 1.1 - Анализ графических данных и поиск блоков

Прецедент 1.2 - Прямое вейвлетное преобразование

Прецедент 1.3 - Децимация графических данных

Прецедент 1.4 - Запись кодированных данных в файл

— Прецедент 2 - Декодировать данные

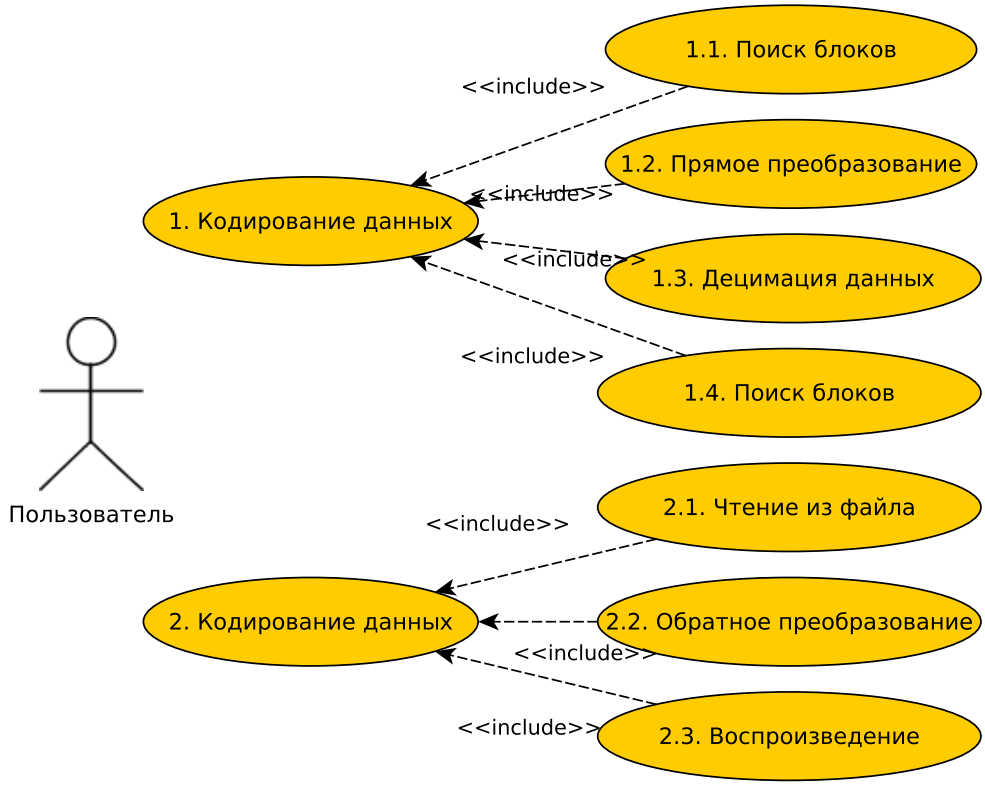

Рисунок 2.1 — Диаграмма вариантов использования с точки зрения пользователя

Прецедент 2.1 - Чтение графических данных из файла

Прецедент 2.2 - Обратное вейвлетное преобразование

Прецедент 2.3 - Воспроизведение видео

2.5 Алгоритм сжатия видео с использованием кратномасштабного анализа

В соответствии со схемой 1.5 прямого и обратного преобразования был разработан алгоритм сжатия видео.

В разрабатываемой системе важна скорость сжатия. Поэтому для разбиения на блоки использовалась усредненная абсолютная разность значений цветовых компонент, так как для ее определения необходимо наименьшее количество вычислений.

Для вейвлетного преобразования необходимо выделить параметры, которые влияют на поведение системы. Были выделены следующие параметры:

— Размер блока

— Порядок системы коэффициентов для вейвлетного преобразования
— Процент сохраняемых данных
— Алгоритм кодирования без потерь

Эти параметры передаются как входные данные вместе с графическими данными.

В разрабатываемой системе для выполнения вейвлет-преобразования необходимо разработать отдельный модуль. Для прямого и обратного преобразований необходимо получить систему вейвлет-коэффициентов нужного порядка. Для этого необходимо разработать функции получения набора коэффициентов указанной системы коэффициентов.

Для алгоритма прямого преобразования необходимо инициализировать размер рассматриваемой области. В начале работы алгоритма размер рассматриваемой области равен размеру блока. В процессе работы алгоритма размер рассматриваемой области уменьшается. Если ширина, высота, глубина рассматриваемой области равны единице, то необходимо закончить вейвлет-преобразование.

На каждой итерации вейвлет-преобразование применяется ко всем векторам рассматриваемой области блока сначала по ширине, затем по высоте, затем по глубине, ширина, высота и глубина рассматриваемой области уменьшаются в два раза. Для выполнения вейвлет-преобразования необходимо составить матрицу вейвлет-преобразования в соответствии с размером рассматриваемой области. Прямое вейвлет-преобразование может быть реализовано как рекурсивно, так и итеративно. Подробная схема разработанного алгоритма прямого вейвлет-преобразования представлена на рисунке 2.2. Код процедуры прямого вейвлетного преобразования блока представлен в листинге 2.1, где `block` - графические данные, которые необходимо преобразовать, `block_size` - количество кадров в блоке, `c` - количество каналов в блоке, `h` - высота блока, `w` - ширина блока, `s` - система коэффициентов, `p` - процент сохраняемых данных.

Листинг 2.1 — Код процедуры transform_block

```
1 transform_block(block, block_size, c, h, w, s, p)
2   block_data ← CopyBlockData(block, block_size, c, h, w)
3   DirectTransform(block_data, c, w, h, block_size, w, h,
      block_size, s)
```

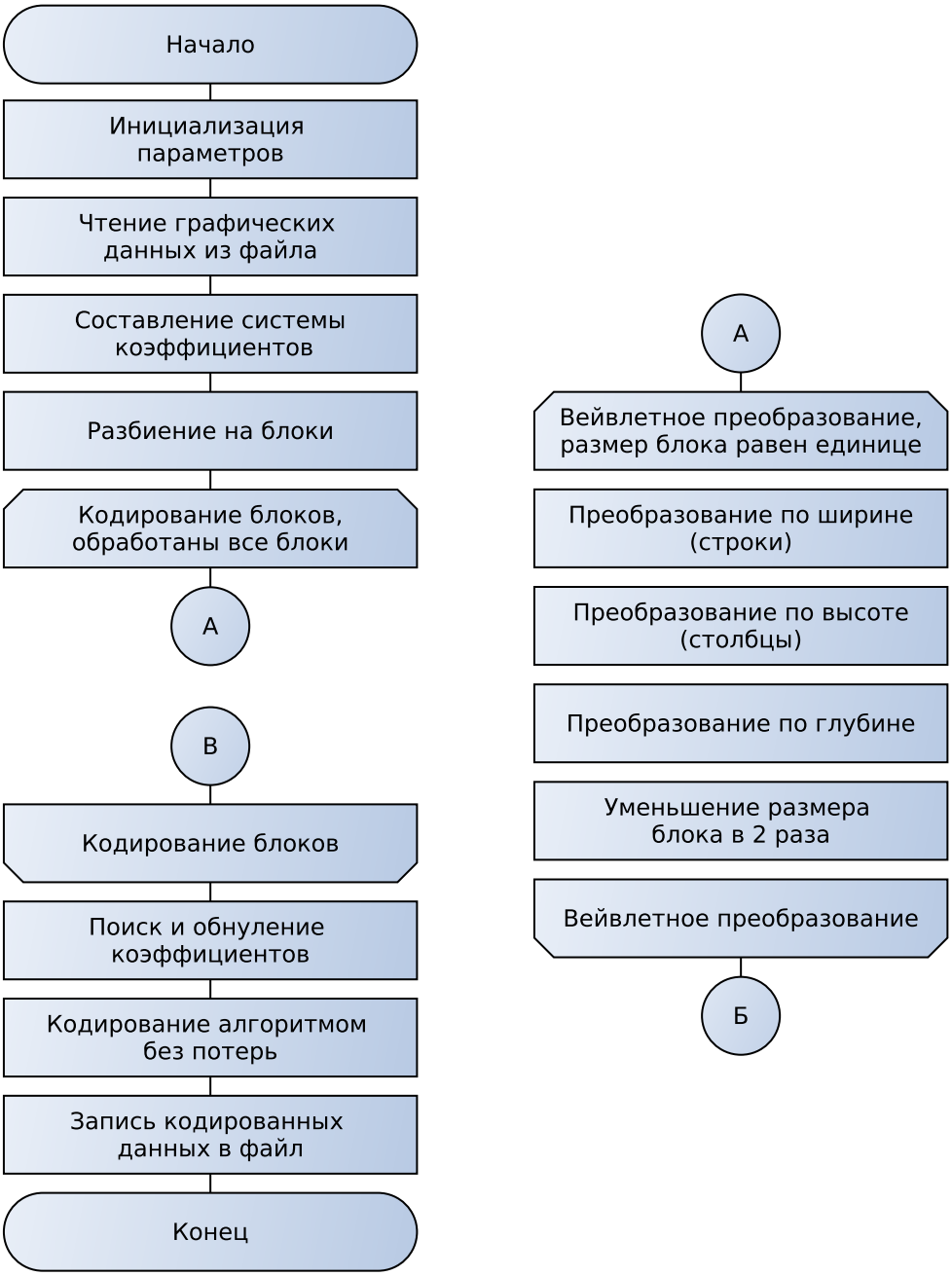

Рисунок 2.2 — Схема компрессии видео

```
4    out_data ← Decimate(block_data, c, w, h, block_size, p)
5    FreeBlockData(block_data, block_size, c, h, w)
6    return out_data
```

Для алгоритма обратного преобразования необходимо рассчитать размер рассматриваемой области, так как блок может иметь разные размеры по высоте, ширине и глубине. В процессе работы алгоритма размер рассматриваемой области увеличивается. Если ширина и высота рассматриваемой области соответствуют размерам изображения, то необходимо закончить обратное вейвлет-преобразование. Подробная схема разработанного алгоритма обратного вейвлет-преобразования представлена на рисунке 2.3. Код процедуры обратного вейвлетного преобразования блока представлен в листинге 2.2, где `out_data` - кодированные данные, к которым необходимо применить обратное вейвлетное преобразование, `block_size` - количество кадров в блоке, `c` - количество каналов в блоке, `h` - высота блока, `w` - ширина блока, `s` - система коэффициентов

Листинг 2.2 — Код процедуры decode_block

```
1  decode_block(out_data, block_size, c, h, w, system)
2    block_data = AllocateBlock(block_size, c, h, w)
3    ConvertArrayToMatrix(out_data, block_data)
4    InverseTransform(block_data, c, w, h, block_size, w, h,
       block_size, system)
5    block = AllocateFrames(block_size, c, h, w)
6    for int i ← 0; i < block_size; ++i
7      block[i] = WriteImageData(block_data[i])
8    FreeBlockData(block_data, block_size, c, h, w)
9    return block
```

2.6 Составление матриц

В данной работе для рассмотрения выбраны вейвлеты Добеши, так как предполагается, что увеличение размера системы коэффициентов приводит к лучшей сжимаемости [1]. Матрица прямого преобразования M состоит из двух прямоугольных матриц L и H, представляющих низкочастотный и высокочастотный фильтры. Матрицы L и H могут быть составлены

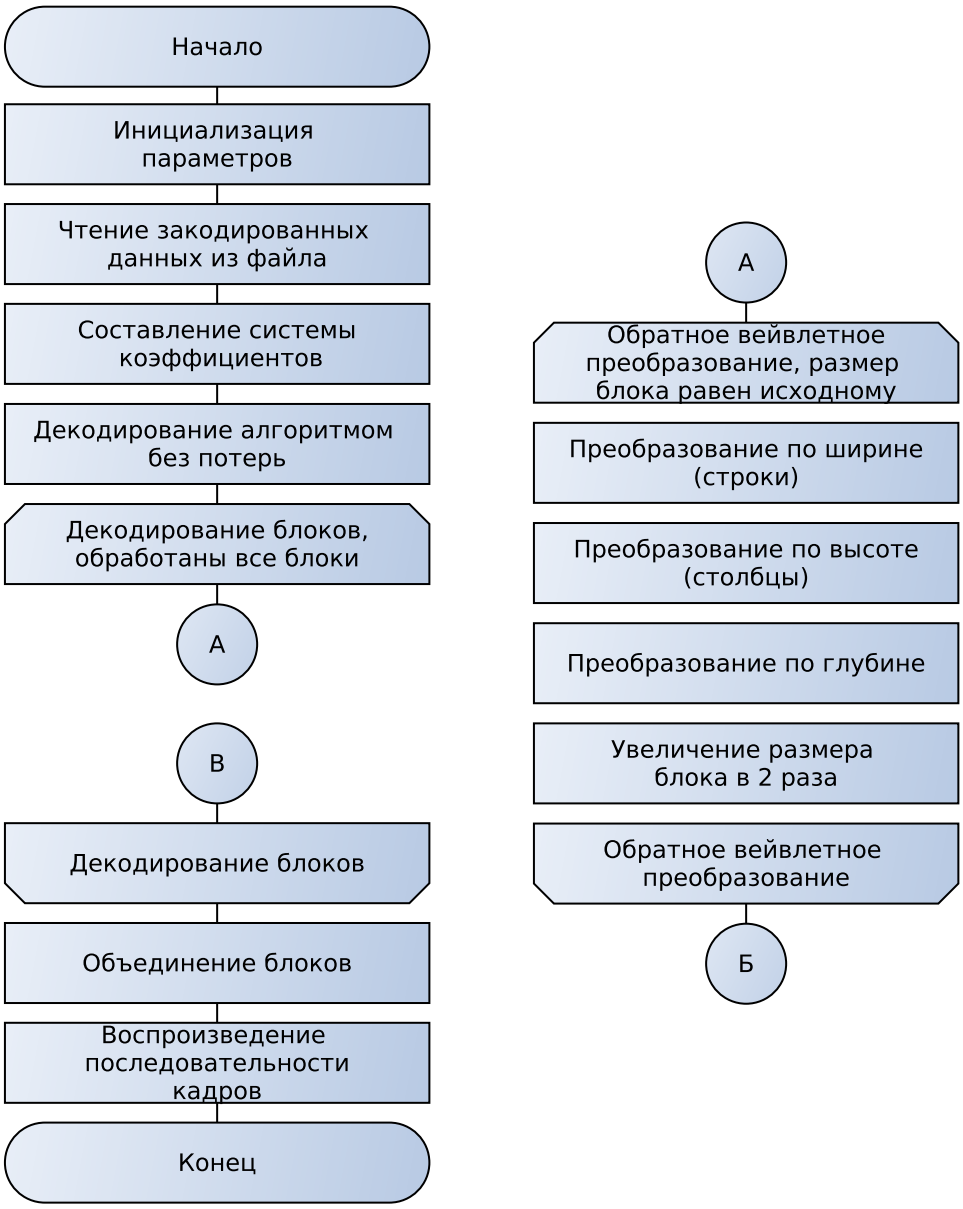

Рисунок 2.3 — Схема компрессии видео

следующим образом:

$$L = \begin{cases} c_{j-2i+1+n}, \text{при } j - 2i + 1 < 0 \\ c_{j-2i+1}, \text{иначе} \end{cases} \tag{2.1}$$

$$L = \begin{cases} (-1)^{j+1} c_{2i-j+n}, \text{при } 2i - j < 0 \\ (-1)^{j+1} c_{2i-j}, \text{иначе} \end{cases} \tag{2.2}$$

Матрица обратного преобразования является транспонированной матрицей прямого преобразования. Код процедуры заполнения матрицы преобразования представлен в листинге 2.3, где m - матрица преобразования, dim - размерность матрицы преобразования, system - выбранная система коэффициентов.

Листинг 2.3 — Код процедуры PrepareMatrix

```
1  PrepareMatrix(m, dim, system)
2    if dim % 2 != 0
3      then dim ← dim - 1
4    i ← 0
5    j ← 0
6    for i ← 0; i < dim / 2; ++i
7      for j ← 0; j < dim; ++j)
8        index ← (j + 1) - 2 * (i + 1) + 1
9        system_index ← index < 0 ? index + dimension : index
10       m[i][j] ← system_index >= 0 && system_index <
             system.size ? system[system_index] : 0
11
12   for i ← 0; i < dim / 2; ++i)
13     for j ← 0; j < dim; ++j
14       index ← 2 * (i + 1) - (j + 1);
15       system_index ← index < 0 ? index + dimension : index
16       if system_index < 0 or system_index >= system.size()
17         then m[i + dim / 2][j] ← 0
18         else
19           if (j + 2) % 2 != 0
20             then m[i + dim / 2][j] = -system[system_index]
21             else m[i + dim / 2][j] = system[system_index]
```

2.7 Поиск и обнуление минимальных коэффициентов после применения вейвлет-преобразования

После применения прямого вейвлетного преобразования необходимо обеспечить сжатие путем обнуления части коэффициентов. Степень сжатия задается пользователем в процентах. Из ненулевых элементов необходимо выбрать указанный пользователем процент максимальных элементов. Это значит, что необходимо найти указанную пользователем порядковую статистику в наборе графических данных. Видео содержит большое количество кадров, каждый кадр содержит несколько каналов, массив графических данных может содержать по несколько тысяч элементов. Поэтому необходимо производить эффективный поиск порядковой статистики. Рассмотрим алгоритм, выполняющий поиск порядковой статистики за линейное время $O(n)$, где n - размерность входных данных для поиска порядковой статистики. Заранее не известно количество ненулевых элементов в массиве, для определения номера искомой порядковой статистики необходимо подсчитать количество ненулевых элементов. Зная это количество и указанный пользователем процент, точно определяется номер искомой порядковой статистики.

Алгоритм `Randomized_Select` поиска порядковой статистики разработан по аналогии с алгоритмом быстрой сортировки. Как и в алгоритме быстрой сортировки, в алгоритме `Randomized_Select` используется идея рекурсивного разбиения входного массива. В отличие от алгоритма быстрой сортировки, в котором рекурсивно обрабатываются обе части разбиение, алгоритм `Randomized_Select` работает лишь с одной частью. Это различие проявляется в результатах анализа обоих алгоритмов: если математическое ожидание времени работы алгоритма быстрой сортировки равно $\Theta(n \lg n)$, то ожидаемое время работы алгоритма `Randomized_Select` равно $\Theta(n)$, в предположении, что все элементы входных множеств различны.

В алгоритме `Randomized_Select` используется процедура Randomized_Partition. Подобно быстрой сортировке, `Randomized_Select` – рандомозированный алгоритм, поскольку его поведение частично определяется выводом генератора случайных чисел. Код процедуры `Randomized_Select` представлен в листинге 2.4, где входные параметры A – Массив данных, p – левая граница части массива для поиска, r –

правая граница части массива для поиска, i – номер искомой порядковой статистики, возвращаемое значение это искомая порядковая статистика.

Листинг 2.4 — Код процедуры Randomized_Select

```
1  Randomized_Select(A, p, r, i)
2    if p = r
3      then return A[p]
4    q ← Randomized_Partition(A, p, r)
5    k ← q − p + 1
6    if i = k
7      then return A[q]
8    else if i < k
9      then return Randomized_Select(A, p, q − 1, i)
10   else return Randomized_Select(A, q + 1, r, i − k)
```

После выполнения процедуры `Randomized_Partition`, которая вызывается в строке 4 представленного выше алгоритма, массив `A[p..r]` оказывается разбитым на два (возможно, пустых) подмассива `A[p..q-1]` и `A[q+1..r]`. При этом величина каждого элемента `A[p..q-1]` не превышает `A[q]`, а величина каждого элемента `A[q+1..r]` больше `A[q]`. Как и в алгоритме быстрой сортировки, элемент `A[q]` будем называть опорным (`pivot`). В строке 5 процедуры `Randomized_Select` вычисляется количество элементов `k` подмассива `A[p..q]`, то есть количество элементов, попадающих в нижнюю часть разбиения плюс один опорный элемент. Затем в строке 6 проверяется, является ли элемент `A[q]` `i`-м в порядке возрастания элементом. Если это так, то возвращается элемент `A[q]`. В противном случае в алгоритме определяется, в каком из двух подмассивов содержится `i`-й в порядке возрастания элемент: в подмассиве `A[p..q-1]` или в подмассиве `A[q+1..r]`. Если `i < k`, то нужный элемент находится в нижней части разбиения, и он рекурсивно выбирается из соответствующего подмассива в строке 9. Если же `i > k`, то нужный элемент находится в верхней части разбиения. Поскольку уже известны `k` значений, которые меньше `i`-го в порядке возрастания элемента массива `A[p..r]` (это элементы подмассива `A[p..q]`), нужный элемент является `(i - k)`-м в порядке возрастания элементом подмассива `A[q + 1..r]`, который рекурсивно находится в строке 10. Опорный элемент выбирается в массиве случайным образом. Код процедуры `Randomized_Partition` представлен в листинге 2.5, где входными

параметрами являются A – массив данных, p – левая граница части массива для поиска, r – правая граница части массива для поиска, возвращаемым значением является индекс опорного элемента.

Листинг 2.5 — Код процедуры Randomized_Partition

```
1 Randomized_Partition(A, p, r)
2   i ← Random( p, r )
3   Swap( A[r], A[i] )
4   return Partition( A, p, r )
```

Вместо того чтобы в качестве опорного элемента всегда использовать `A[r]`, такой элемент будет выбираться в массиве `A[p..r]` случайным образом. Процедура `Partition` выбора опорного элемента представлена в листинге 2.6.

Листинг 2.6 — Код процедуры Partition

```
1 Partition(A, p, r)
2   x ← A[r]
3   i ← p − 1
4   for j ← p to r − 1
5     do if A[j] <= x
6       then i ← i + 1
7         Swap( A[i], A[j] )
8   Swap( A[i + 1], A[r] )
9   return i + 1
```

Рассмотренный алгоритм выполняет поиск порядковой статистики за линейное время [3].

2.8 Процессы, происходящие в разрабатываемой системе

В разрабатываемой системе происходят следующие процессы:

— чтение и запись графических данных в файл,
— операции выделения и освобождения памяти для вычислений,
— анализ графических данных для определения блоков,
— прямое и обратное вейвлет-преобразование данных,
— операции с матрицами для работы вейвлет-преобразования,
— состевление матрицы коэффициентов для вейвлет-преобразования,

— удаление части информации о видео после применения вейвлет-преобразования.

В результате процессов, происходящих в разрабатываемой системе, была разработана схема основных модулей, необходимых для функционирования системы, представленная на рисунке 2.4.

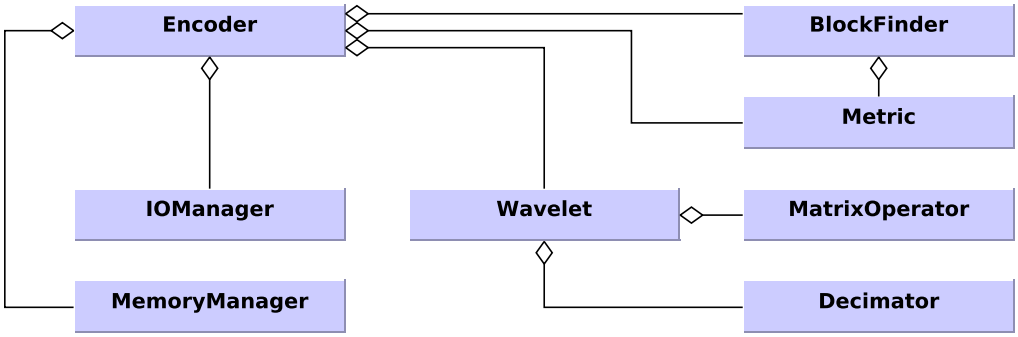

Рисунок 2.4 — Схема модулей разрабатываемой системы

Модуль `Encoder` является основным модулем разрабатываемой системы, в котором происходит весь процесс обработки графических данных от чтения графических данных из файла до записи закодированных графических данных в файл. Содержит функции:

— функция кодирования исходных графических данных,
— функция декодирования преобразованных графических данных.

Модуль `BlockFinder` вычисляет блоки, к которым будет применено вейвлетное преобразование. Содержит функцию поиска блоков, принимающую на вход графические данные и возвращающую вычисленные блоки.

Модуль `IOManager` чтения/записи данных в файл при чтении получает от управляющего модуля имя файла и возвращает прочитанные графические данные, при записи получает от управляющего модуля имя файла и графические данные и возвращает результат выполнения операции записи. Содержит следующие функции:

— функция чтения графических данных из файла,
— функция записи преобразованных графических данных в файл,
— функция чтения преобразованных графических данных из файла.

Модуль `MemoryManager` управляет памятью, необходимой для выполнения расчетов во время применения вейвлетного преобразования. Содержит следующие функции:

— функция выделения блока памяти для расчетов,

— функция освобождения выделенной памяти,

— функция копирования графических данных во временную память для применения прямого вейвлетного преобразования,

— функция копирования преобразованых данных во временную память для применения обратного вейвлетного преобразования.

Модуль вейвлетных преобразований `Wavelet` получает на вход графические данные и параметры преобразования, применяет к данным указанное преобразование, возвращает преобразованные данные. Содержит следующие функции:

— функция выполнения прямого вейвлетного преобразования,

— функция выполнения обратного вейвлетного преобразования.

Для работы модуля вейвлетных преобразований требуется вспомогательный модуль работы с матрицами `MatrixOperator`, который производит умножение матрицы на вектор, а так же составление матриц для указанного вейвлетного преобразования. Содержит следующие функции:

— функция составления матрицы преобразования указанной размерности для указанной системы коэффициентов,

— функция получения набора коэффициентов указанной системы коэффициентов,

— функция умножения матрицы на вектор графических данных,

— функция транспонирования матрицы.

Алгоритм составления матрицы представлен на рисунке

Модуль выбора значимых коэффициентов `MatrixOperator` получает от модуля `Wavelet` графические данные и степень сжатия, удаляет указанную часть данных, возвращает обработанные данные. Содержит следующие функции:

— функция удаления части графических данных,

— функция поиска порядковой статистики.

3 Технологический раздел

3.1 Выбор языка программирования

В настоящей работе для реализации проекта был выбран язык программирования C++ по следующим причинам:

— является языком высокого уровня, что позволяет разрабатывать программное обеспечение на высоком уровне абстракций,

— обладает преимуществами разработки программного обеспечения при использовании объектно-ориентированного подхода, обеспечивает свойства объектно-ориентированного подхода: инкапсуляцию, наследование, полиморфизм,

— предоставляет удобные средства для обработки исключительных ситуаций,

— обладает набором стандартных коллекций (вектор, список, и другие),

— является кроссплатформенным.

Для разработки и отладки приложения была выбрана среда разработки Visual Studio 2010. Эта среда разработки обладает удобным и интуитивно понятным интерфейсом пользователя, удобными средствами разработки и отладки приложений, а так же встроенными средствами для разработки GUI.

3.2 Использование сторонних библиотек

В разрабатываемом проекте необходимо производить обработку видео, выбранных пользователем. Так как существует большое количество форматов представления графических данных, для сокращения времени разработки использовалась библиотека OpenCV, которая предоставляет интерфейс для чтения и записи графических данных в файлы основных используемых форматов. Библиотека OpenCV является свободно распространяемой, имеет обширную документацию, постоянно обновляется, что является ее преимуществами перед другими библиотеками. Для работы прямого и обратного вейвлет-преобразований важна скорость работы. Так как вейвлет-преобразование состоит в умножении строк матрицы графи-

ческих данных на матрицу преобразования, эту операцию можно распределить по потокам. Для распределения операций по потокам использовался открытый стандарт для распараллеливания программ на языке C++ OpenMP. Наличие инструментов OpenMP в среде разработки Visual Studio 2010 также упрощает ее использование. Стандарт OpenMP является свободно распространяемым, имеет обширную документацию, постоянно обновляется, что является его преимуществами перед другими аналогичными инструментами.

3.3 Требования к программной совместимости

Разработка и тестирование программного обеспечения веласть на компьютере с установленной операционной системой Windows 7. Для корректной работы исполняемого файла необходимо наличие библиотек OpenCV в одной директории с исполняемым файлом. Минимальные Требования к компьютеру для запуска разработанного программного обеспечения указаны в таблице 3.1.

Таблица 3.1 — Минимальные требования к техническим характеристикам компьютера

Характеристика	Требуемое значение
Процессор	Intel Core 2 Duo CPU T5800 2.00GHz x2
Оперативная память	2 Гб
Операционная система	Windows 7 x32
Жесткий диск	1Гб свободного дискового пространства

3.4 Разработанные классы в программном комплексе

В соответствии со схемой модулей 2.4 была разработана структура классов, изображенная на рисунке 3.1.

Класс `Encoder` является точкой входа в разрабатываемую систему. Содержит методы:

— `void Encode(char *fileName, int rate);`
— `void Decode(char *fileName);`

В метод `Encode` передаются имя входного файла с графическими данными и коэффициент децимации. Метод выполняет анализ графиче-

ских дынных, применяет вейвлетное преобразование к графическим данным, проводит децимацию и кодирование коэффициентов, осуществляет запись кодированных коэффициентов в файл. В метод `Decode` передается имя входного файла с закидированными данными. Метод проводит чтение закодированных коэффициентов из файла, применяет обратное вейвлетное преобразование, составляет из преобразованых блоков кадры и проводит воспроизведение последовательности кадров.

Класс `Wavelet` вейвлетных преобразований получает на вход графические данные и параметры преобразования, применяет к данным указанное преобразование, возвращает преобразованные данные. Содержит следующие методы:

— `std::vector<int> transform_block(IplImage **block, int block_size, int nChannels, int height, int width, int system, int savePerc)` - метод прямого вейвлетного преобразования. Параметры: графические данные `block`, размер блока `block_size`, количество каналов `nChannels`, ширина `width` кадра, высота `height` кадра, система коэффициентов преобразования `system`

— `IplImage ** decode_block(std::vector<int> out_data, int block_size, int nChannels, int height, int width, int system)` - функция выполнения обратного вейвлет-преобразования. Параметры: графические данные `out_data`, размер блока `block_size`, количество каналов `nChannels`, ширина `img_width` кадра, высота `img_height` кадра, система коэффициентов преобразования `system`.

— `void DirectTransform(float ****image_data, int nChannels, int img_width, int img_height, int img_depth, int width, int height, int depth, int transform_type);` - метод прямого вейвлетного преобразования блока. Параметры: графические данные блока `image_data`, количество каналов `nChannels`, ширина `img_width` блока, высота `img_height` блока, глубина `img_depth` блока, ширина `width` рассматриваемой области, высота `height` рассматриваемой области, глубина `depth` рассматриваемой области, система коэффициентов преобразования `transform_type`,

— `void InverseTransform(float ****image_data, int nChannels, int img_width, int img_height, int img_depth, int`

width, int height, int depth, int transform_type); - - метод обратного вейвлетного преобразования блока. Параметры: графические данные блока image_data, количество каналов nChannels, ширина img_width блока, высота img_height блока, глубина img_depth блока, ширина width рассматриваемой области, высота height рассматриваемой области, глубина depth рассматриваемой области, система коэффициентов преобразования transform_type,

— void DirectTransformIteration(float ****image_data, int channel_index, int part_count, int index1, int index2, int direction, const std::vector<std::vector<float> > &matrix_ad); – метод выполнения итерации прямого вейвлет-преобразования. Параметры: графические данные image_data, номер канала channel_index, длина вектора для преобразования part_count, номера векторов для преобразования в двух измерениях index1 и index2, направление direction, матрица преобразования matrix_ad

— void InverseTransformIteration(float ****image_data, int channel_index, int part_count, int index1, int index2, int direction, const std::vector<std::vector<float> > &matrix_ad); – функция выполнения итерации обратного вейвлет-преобразования. Параметры: графические данные image_data, номер канала channel_index, длина вектора для преобразования part_count, номера векторов для преобразования в двух измерениях index1 и index2, направление direction, матрица преобразования matrix_ad.

Метод transform_block был разработан в соответствии со схемой прямого вейвлетного преобразования, представленной на рисунке 2.2. Код метода представлен в литинге 3.1. Метод decode_block выполняет схожие действия в обратном порядке.

Листинг 3.1 — Код метода transform_block

```
1 std::vector<int> transform_block(IplImage **block, int block_size, int nChannels, int height,
      int width, int system, int savePerc){
2   float ****block_data = new float***[block_size];
3   for (int i = 0; i < block_size; ++i){
4     float ***image_data = NULL;
5     image_data = AllocateImageMemory(nChannels, height, width, image_data);
6     auto frame = block[i];
7     CopyImageData(frame, image_data, nChannels, height, width);
8     block_data[i] = image_data;
9   }
```

```
10
11    DirectTransform(block_data, nChannels, width, height, block_size, width, height,
          block_size, system);
12    std::vector<int> out_data = Decimate(block_data, nChannels, width, height, block_size,
          savePerc);
13    for (int i = 0; i < block_size; ++i)
14      FreeImageMemory(nChannels, height, width, block_data[i]);
15    delete[] block_data;
16    return out_data;
17  }
```

Класс `BlockFinder` содержит метод поиска блоков на последовательности кадров: `IplImage ** FindBlocks(IplImage *frames, int framesSize);`, где `frames` - последовательность кадров, `framesSize` - длина последовательности кадров.

Класс `Metrix` служит для поиска границ блоков, а так же для оценки качества видео после применения алгоритмов кодирования. Содержит методы поиска разности между кадрами и между последовательностями кадров:

— `float CompareFrames(IplImage &frame1, IplImage &frame2);`, где `frame1` и `frame2` - сравниваемые кадры,

— `float CompareFiles(IplImage *frames1, IplImage *frames2);`, где `frames1` и `frames2` - сравниваемые файлы.

Класс `MatrixOperator` производит умножение матрицы на вектор, а так же составление матриц для указанного вейвлет-преобразования. Содержит следующие методы:

— `void PrepareMatrix(std::vector<std::vector<double> > &matrix, int dimension, const std::vector<double> &system)` - функция составления матрицы преобразования указанной размерности для указанной системы коэффициентов Параметры: матрица преобразования `matrix`, размерность матрицы, `dimension`, система коэффициентов system.

— `std::vector<double> PrepareSystem(int transform_type)` - функция получения набора коэффициентов указанной системы коэффициентов Параметры: тип преобразования `transform_type`.

— `void MultiplyMatrixVector(const std::vector<std::vector<double> > &matrix, double ***image_data, int channel_index, int`

part_count, int vector_index, int direction) - функция умножения матрицы на вектор графических данных Параметры: матрица преобразования matrix, графические данные image_data, номер канала channel_index, длина вектора для преобразования part_count, номер вектора для преобразования vector_index, направление (строка или столбец) direction.

— void TransposeMatrix(std::vector<std::vector<double> > &matrix) - функция транспонирования матрицы Параметры: матрица преобразования matrix.

Класс Decimator выбора значимых получает от управляющего модуля графические данные и степень сжатия, удаляет указанную часть данных, возвращает обработанные данные. Содержит методы:

— std::vector<int> Decimate(float ****image_data, int nChannels, int width, int height, int depth, char decimatio_ratio); – функция удаления части графических данных. Параметры: графические данные image_data, количество каналов nChannels, ширина width, высота height, глубина depth блока, процент сохраняемых данных decimation_ratio.

— float RandomizedSelect(std::vector<float> &numbers, int left, int right, int level) - метод поиска порядковой статистики, где numbers - рассматриваемый набор данных, left - левая граница части массива для поиска, right – правая граница части массива для поиска, level – номер искомой порядковой статистики,

— int Partition(std::vector<float> &numbers, int left, int right); - метод выбора опорного элемента случайным образом. Параметры: numbers - рассматриваемый набор данных, left - левая граница части массива для поиска, right – правая граница части массива для поиска,

— int RandomizedPartition(std::vector<float> &numbers, int left, int right); - метод выбора опорного элемента. Параметры: numbers - рассматриваемый набор данных, left - левая граница части массива для поиска, right – правая граница части массива для поискаю

Класс IOManager производит чтение и запись в файл закодированных графических данных. Содержит методы:

— `void ReadData(char file_name[])` – функция чтения графических данных из файла. Параметры: имя файла `file_name`.

— `void WriteEncodedData(char file_name[], std::vector<int> data)` – функция записи кодированных данных в файл. Параметры: имя файла `file_name`, данные `data`.

— `std::vector<int> ReadEncodedData(char file_name[])` – функция чтения кодированных данных из файла. Параметры: имя файла `file_name`.

Формат файла с закодированными данными представлен на рисунке 3.2, где NChannels - количество каналов для представления цвета, Width - ширина кадра, Heigth - высота кадра, BlockLength - длина очередного блока, BlockData - графические данные очередного блока.

Класс `MemoryManager` предназначен для управления памятью для графических данных. Содержит методы:

— `double*** AllocateImageMemory(int channels_count, int row_count, int column_count, double ***image_data)` – функция выделения памяти для данных о кадре Параметры: количество каналов `channels_count`, количество строк `row_count`, количество столбцов `column_count`, данные о кадре `image_data`.

— `void FreeImageMemory(int channels_count, int row_count, int column_count, double ***image_data)` – функция освобождения памяти для данных о кадре Параметры: количество каналов `channels_count`, количество строк `row_count`, количество столбцов `column_count`, данные о кадре `image_data`.

— `void CopyImageData(IplImage *in_image, double ***image_data)` – функция копирования графических данных из кадра во внутреннюю структуру данных Параметры: входной кадр `in_image`, данные о кадре `image_data`.

— `void WriteImageData(double ***image_data, IplImage *in_image)` – функция копирования графических данных из внутренней структуры данных в кадр. Параметры: кадр `in_image`, данные о кадре `image_data`.

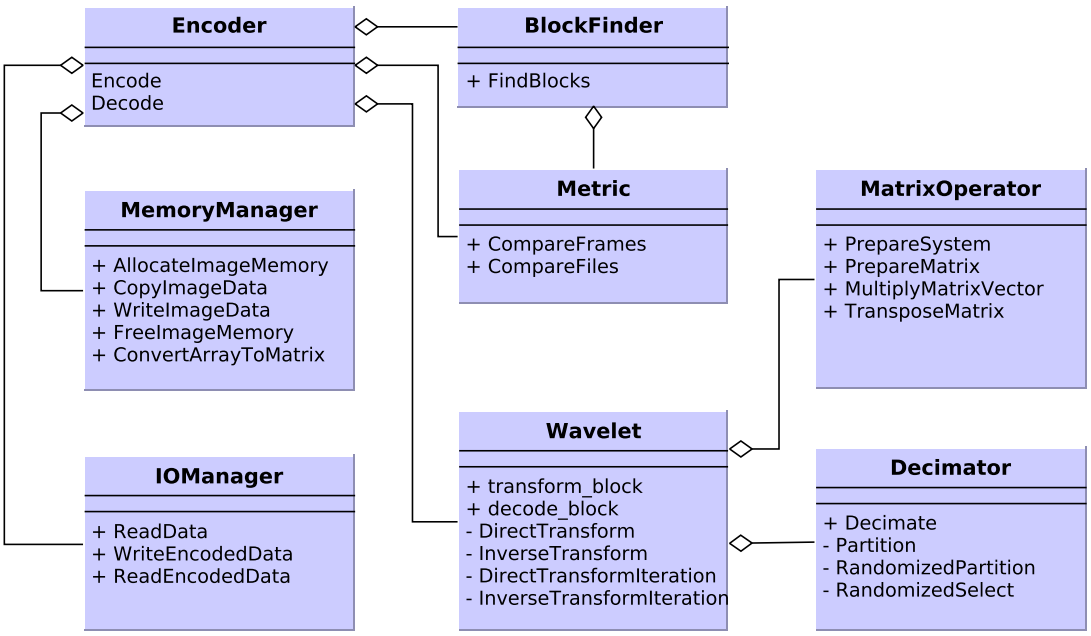

Рисунок 3.1 — Диаграмма классов разрабатываемой системы

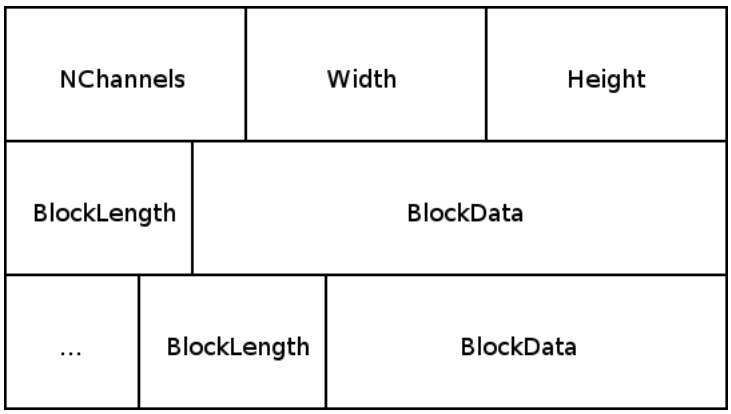

Рисунок 3.2 — Формат файла с закодированными графическими данными

— `void ConvertArrayToMatrix(std::vector<int> &data, double ***image_data)` – функция преобразования считанного кодированного массива в матрицу графических данных кадра Параметры: массив данных `data`, данные о кадре `image_data`.

3.5 Кадры, получаемые при применении вейвлетного преобразования

Пример кадра из видео, к которому было применено вейвлетное преобразование, представлен в приложении Б. Вейвлет-преобразование выполнено с сохранением 10% (вверху справа), 5% (внизу слева), 2% (внизу справа) коэффициентов. Вверху слева представлено исходное изображение. На представленном изображении видно, что при сохранении большего количества коэффициентов наблюдается эффект сглаживания однородных областей. При уменьшении количества сохраненяемых коэффициентов на изображении наблюдается появление артефактов. Мелкие детали, представленные на исходном изображении, при высокой степени сжатия размываются, образуя пятна на получаемом изображении.

3.6 Пользовательский интерфейс разрабатываемого программного комплекса

Пользовательский интерфейс разрабатываемого программного комплекса представлен на рисунке 3.3. В соответствии с разработанными сценариями функционирования программного комплекса был спроектирован пользовательских интерфейс.

Предоставлена возможность выбора исходного файла с графическими данными. При нажатии на кнопку выбора файла пользователю предлагается стандартный диалог выбора файла, принятый в разрабатываемой операционной системе. После выбор файла имя выбранного файла отображается в текстовом поле.

Ползователю предоставляется возможность указать процент сохраняемых данных в текстовом поле. Введенное число должно быть целым в диапазоне от 1 до 99.

При нажатии на кнопку «Кодировать» выполняется поиск блоков, прямое вейвлетное преобразование блоков, децимация графических дан-

ных и запись закодированных данных в файл. При нажатии на кнопку «Декодировать» выполняется чтение из файла, обратное вейвлетное преобразование, воспроизведение графических данных.

При отсутствии выбранного файла или при ошибочном вводе процента сохраняемых данных отображается сообщение об ошибке.

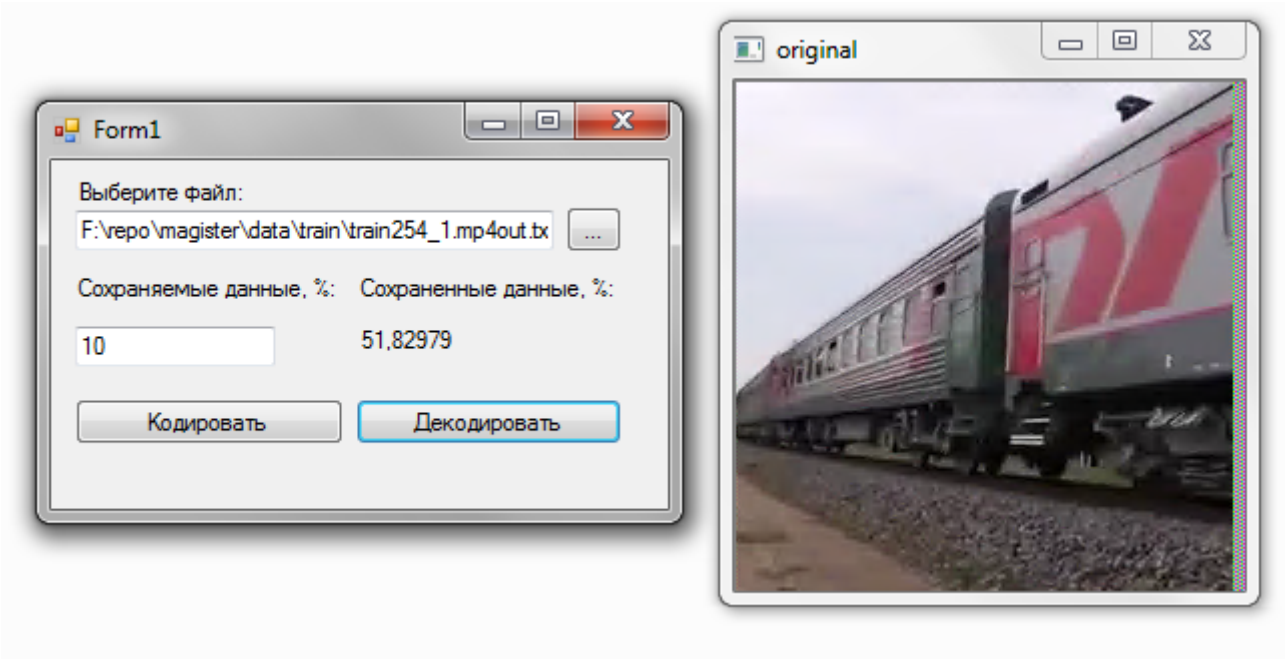

Рисунок 3.3 — Пользовательский интерфейс разрабатываемого программного комплекса

4 Экспериментальный раздел

4.1 Модель данных

Как было описано ранее, при сравнении алгоритмов сжатия видео необходимо учитывать следующие особенности видео:

— качество различных кадров одного и того же видео после обработки одним алгоритмом отличается
— алгоритмы сжатия видео создаются для различных типов видео
— качество сжатия видео зависит от параметров кодирования

Для учета этих особенностей необходимо провести классификацию графических данных на видео. В настоящей работе была разработана модель $M = (V, S, C, B, T)$. Были выделены следующие характеристики:

— V - Количество движения (съемка движущихся объектов, помещения с камеры наблюдения)
— S - Количество однотонных областей (реальная съемка, мультипликация)
— C - Цветовая характеристика (цветное, черно-белое видео)
— B - Битрейт видео
— T - Длительность видео

Для исследования работы метода необходимо кодировать видео из различных классов при фиксированном одном из параметром и варьировании других. Были выбраны видео следующего типа:

— съемка с камеры наблюдения
— съемка движения поезда
— мультипликация с движением
— мультипликация без движения

Каждый тип видео представлен в цветном и черно-белом формате при разрешениях 100×100, 140×140, 175×175, 200×200, 225×225 точек и длительности 25, 50, 75, 100, 125 кадров. Таким образом для исследования было подготовлено 100 файлов. Перевод в градации серого осуществлялся программно, что увеличивает размер набора входных данных до 200.

4.2 Степень сжатия видео в зависимости от количества движения на видео

Для оценки зависимости степени сжатия видео от количества движения с помощью разработанного метода были выбраны два типа файлов, содержащие реальную съемку. Это видео, содержащие непрерывное движение, и видео с неподвижным объектом. Для оценки непрерывного движения была выбрана съемка движения поезда, где движение происходит по площади всего кадра в течение всего видео. Для оценки отсутствия движения было выбрано видео, содержащее съемку неподвижного вагона поезда, так как такие данные соответствуют по количеству однотонных областей выбранному видео с движением.

Ожидалось, что отсутствие движения позволит сильнее сжать видео, так как при отсутсвии движения возникает избыточность графических данных по времени. Не происходит смен сцен, поэтому блок может быть выбран таким образом, чтобы содержать максимальное количество кадров. Вейвлетное преобразование такого блока позволит обнулять большее количество полученных коэффициентов после применения вейвлетного преобразования. Было проведено сравнение видео файлов, отличающихся по длительности и разрешению. Для файлов с одинаковым разрешением, но различной длительностью, было расчитано среднее значение степени сжатия.

График зависимости процента сохраняемых данных от длительности для видео с движением и видео без движения представлен на рисунке 4.1.

В результате эксперимента было обнаружено, что степень сжатия видео без движения превосходит степень сжатия видео с движением.

4.3 Степень сжатия видео в зависимости от количества однотонных областей на видео

Для оценки зависимости степени сжатия видео от количества однотонных областей с помощью разработанного метода были выбраны два типа файлов. Это видео, содержащие реальную съемку, и видео содержащие мультипликацию. Оба типа видео были выбраны без движения. Муль-

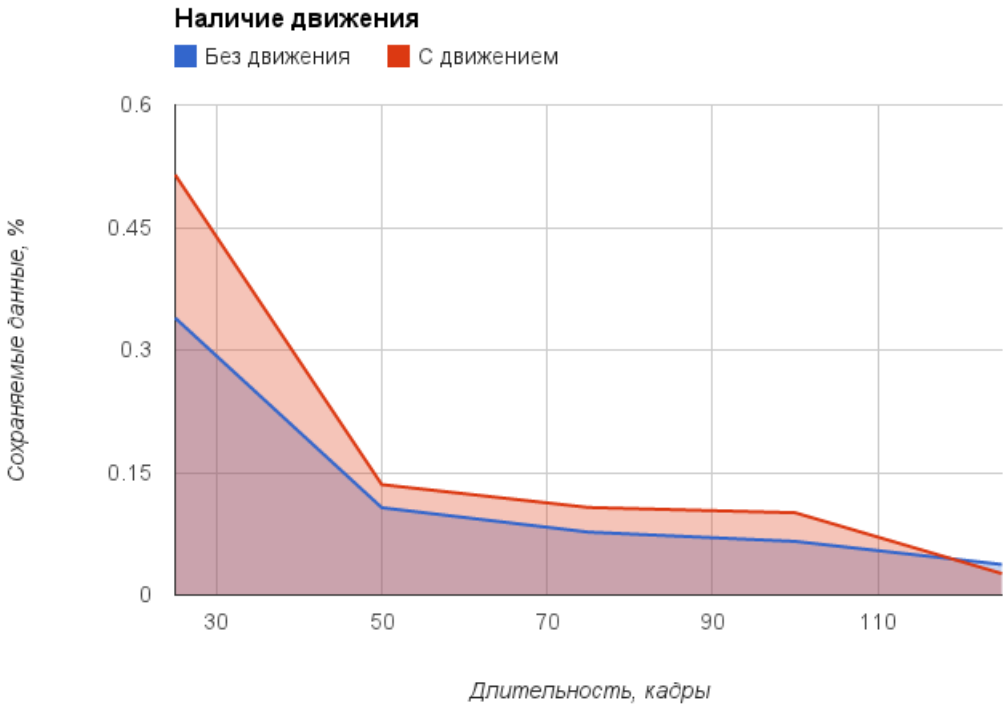

Рисунок 4.1 — Степень сжатия видео в зависимости от количества движения на видео

типликация содержит большее количество однотонных областей, так как при раскраске кадра часто используется заливка. Реальная съемка содержит большое количество мелких деталей, что вызывает необходимость сохранять большее коичество коэффициентов после применения вейвлетного преобразования для сохранения границ.

Ожидалось, что мультипликация позволит сильнее сжать видео, вейвлетное преобразование имеет преимущество при наличии большого количества однотонных областей на кадре. В выбраных видео нет движения, не происходит смен сцен, поэтому размер блока не влияет на степень сжатия при выборе одинакового значения для обоих типов видео. Наличие большого количество однотонных областей позволит удалить большее количество коэффициентов, полученных после вейвлетного преобразования. Было проведено сравнение видео файлов, отличающихся по длительности и разрешению. Для файлов с одинаковым разрешением, но различной длительностью, было расчитано среднее значение степени сжатия.

График зависимости процента сохраняемых данных от наличия мультипликации для видео без движения представлен на рисунке 4.2.

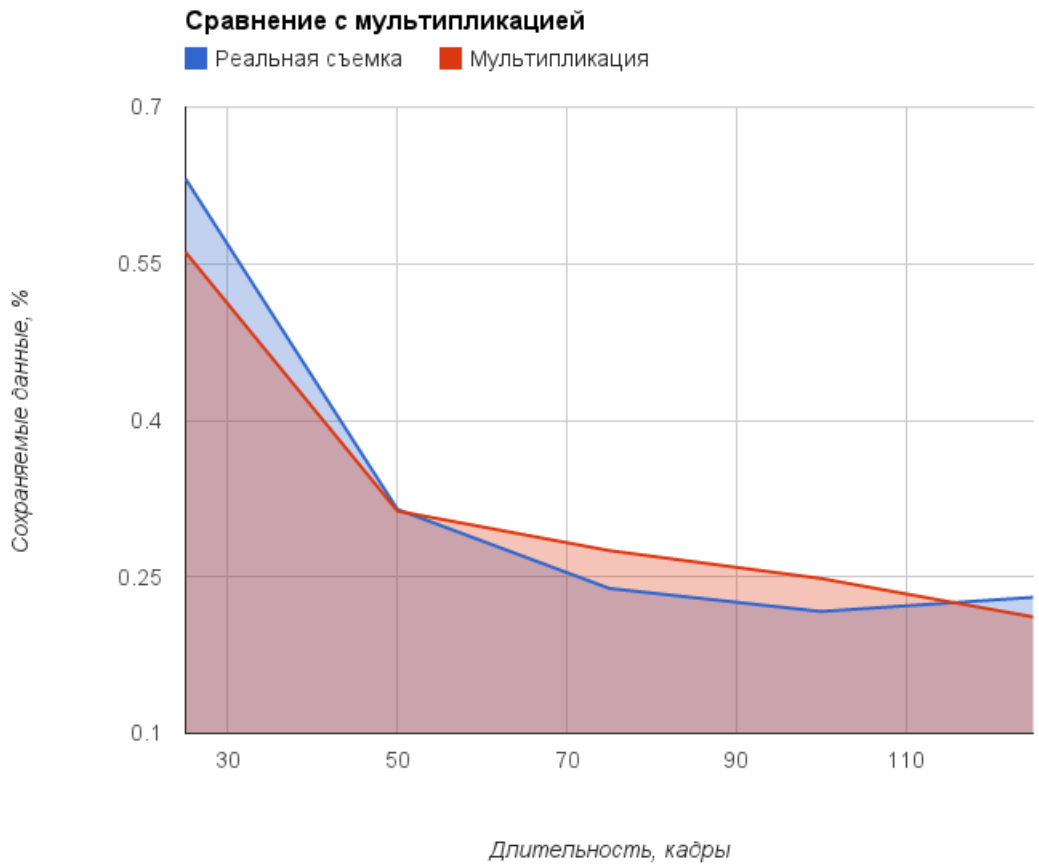

Рисунок 4.2 — Степень сжатия видео в зависимости от количества однотонных областей на видео

В результате эксперимента было обнаружено, что мультипликация не дает увеличения при сжатии. Это объясняется тем, что современные мультипликации становятся все более похожими на реальную съемку, так как при создании мультипликаций все чаще используются средства вычислительной техники, позволяющие создавать фотореалистичные изображения. Растет количество деталей на каждом кадре, увеличивается количество границ на кадре. Это не дает преимуществ при использовании вейвлетного преобразования.

4.4 Степень сжатия видео в зависимости от цветовых характеристик видео

Для оценки зависимости степени сжатия видео от цветовых характеристик видео с помощью разработанного метода были выбраны два типа

файлов. Это видео, представленные с помощью цветовой модели RGB, и те же видео после удалении информации о цвете. Видео в градациях серого содержат большее количество смещных пикселей с одинаковым значением, чем видео с подробной информацией о цвете. При переводе видео в градации серого смежные пиксели, отличающиеся цветом, но имеющие одинаковую яркость, имеют одинаковое значение. вейвлетного преобразования.

Ожидалось, что отсутсвие информации о цвете позволит сильнее сжать видео, так как вейвлетное преобразование имеет преимущества при наличии большого количества однотонных областей на кадре. В выбраных видео одинаковое количество движения, поэтому размер блока не влияет на степень сжатия при выборе одинакового значения для обоих типов видео. Было проведено сравнение видео файлов, отличающихся по длительности и разрешению. Для файлов с одинаковым разрешением, но различной длительностью, было расчитано среднее значение степени сжатия.

График зависимости процента сохраняемых данных от цветовых характеристик видео представлен на рисунке 4.3.

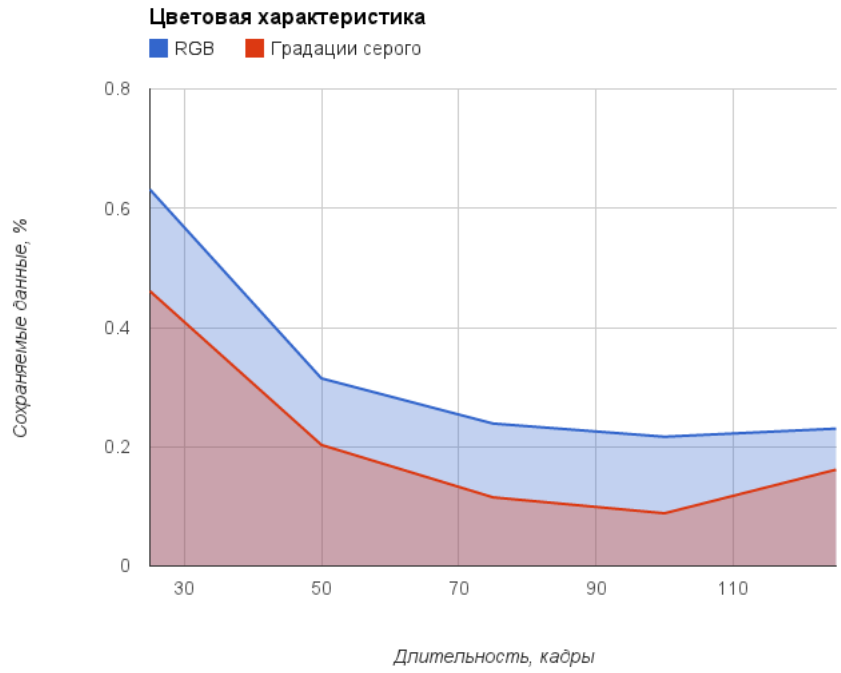

Рисунок 4.3 — Степень сжатия видео в зависимости от цветовых характеристик видео

В результате эксперимента было обнаружено, что отсутствие информации о цвете позволяет повысить степень сжатия.

4.5 Анализ результатов экспериментов

В результате экспериментов были определены типы графических данных, применения к которым разработанного медода дает большую степень сжатия. Результаты экспериментов представлены в таблице 4.1. Было определено, что разработанный метод имеет преимущества при применении к видео, имеющим наименьшее количество движения, наименьшее количество информации о цвете и наибольшее количество однотонных областей.

Таблица 4.1 — Анализ результатов экспериментов

Характеристика M	Преимущественный тип данных	Причина
Количество движения V	Видео без движения	Отсутствие движения в блоке позволяет использовать избыточность данных в глубину
Количество однотонных областей S	Не выявлен	Современные мультипликации становятся близки к реальной съемке
Цветовая характеристика C	Видео в градациях серого	Пиксели разного цвета близкой яркости при удалении цветовой информации близки по значению, образуются однотонные области

Заключение

В результате работы был разработан метод сжатия видео на основе кратномасштабного анализа. Были выполнены следующие задачи:

— проведен анализ существующих алгоритмов сжатия видео,

— разработан метод сжатия видео на основе кратномасштабного анализа,

— создано приложение, позволяющее применять предложенный метод к видео, предоставляющее возможности: расчет размеров блоков для вейвлетного преобразования, применение к видео прямого и обратного вейвлет-преобразования, выбор данных для сохранения после вейвлет-преобразования, расчет коэффициента сжатия видео, измерение времени работы прямого вейвлет-преобразования, обратного вейвлет-преобразования, выбора данных для сохранения после вейвлет-преобразования,

— подготовлена модель графических данных для исследования работы метода,

— проведено исследование работы метода, исследована степень сжатия видео в зависимости от типа графических данных,

— определено, что метод применим для данных в градациях серого и без движения.

Список использованных источников

1. *Добеши, И.* Десять лекций по вейвлетам / И. Добеши. — Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.

2. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. — М.: ДИАЛОГ-МИФИ, 2003.

3. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. — М.: Издательский дом «Вильямс», 2005.

4. *Артюшенко, В.М.* Цифровое сжатие видеоинформации и звука / В.М. Артюшенко, О.И. Шелухин, М.Ю. Афонин. — М.: Издательско-торговая корпорация «Дашков и Ко», 2003.

5. *Уэлстид, С.* Фракталы и вейвлеты для сжатия изображений в действии / С. Уэлстид. — М.: Издательство Триумф, 2003.

6. *Тропченко, А.Ю*. Методы сжатия изображений, аудиосигналов и видео / А.Ю. Тропченко, А.А. Тропченко. — СПб.: ИТМО Государственный Университет, 2009.

7. *Воробьев, В.И.* Теория и практика вейвлет-преобразования / В.И. Воробьев, В.Г. Грибунин. — СПб.: Типография ВУС, 1999.

8. *Столниц, Э.* Вейвлеты в компьютерной графике, теория и приложения / Э. Столниц, Т. ДеРоуз, Д. Салезин. — М.: Ижевск, 2002.

9. *Стройкова, К.А.* Применение вейвлетного преобразования для сжатия изображений / К.А. Стройкова, О.В. Рогозин. — М.: Издательство МГТУ им. Н.Э. Баумана, 2013.

10. *Стройкова, К.А.* Особенности обработки растровых изображений на основе дискретного вейвлет-преобразования / К.А. Стройкова, О.В. Рогозин. — М.: Издательство МГТУ им. Н.Э. Баумана, 2013.

Приложение А Системы коэффициентов Добеши

Таблица А.1 — Ортогональные нормированные коэффициенты Добеши низких порядков

D2	D4	D6	D8	D10	D12	D14	D16	D18	D20
1	0.6830127	0.47046721	0.32580343	0.22641898	0.15774243	0.11009943	0.07695562	0.05385035	0.03771716
1	1.1830127	1.14111692	1.01094572	0.85394354	0.69950381	0.56079128	0.44246725	0.34483430	0.26612218
	0.3169873	0.650365	0.8922014	1.02432694	1.06226376	1.03114849	0.95548615	0.85534906	0.74557507
	-0.1830127	-0.19093442	-0.03957503	0.19576696	0.44583132	0.66437248	0.82781653	0.92954571	0.97362811
		-0.12083221	-0.26450717	-0.34265671	-0.31998660	-0.20351382	-0.02238574	0.18836955	0.39763774
		0.0498175	0.0436163	-0.04560113	-0.18351806	-0.31683501	-0.40165863	-0.41475176	-0.35333620
			0.0465036	0.10970265	0.13788809	0.1008467	6.68194092e-4	-0.13695355	-0.27710988
			-0.01498699	-0.00882680	0.03892321	0.11400345	0.18207636	0.21006834	0.18012745
				-0.01779187	-0.04466375	-0.05378245	-0.02456390	0.043452675	0.13160299
				4.71742793e-3	7.83251152e-4	-0.02343994	-0.06235021	-0.09564726	-0.10096657
					6.75606236e-3	0.01774979	0.01977216	3.54892813e-4	-0.04165925
					-1.52353381e-3	6.07514995e-4	0.01236884	0.03162417	0.04696981
						-2.54790472e-3	-6.88771926e-3	-6.67962023e-3	5.100433697e-3
						5.00226853e-4	-5.54004549e-4	-6.05496058e-3	-0.01517900
							9.55229711e-4	2.61296728e-3	1.97332536e-3
							-1.66137261e-4	3.25814671e-4	2.87768659e-3
								-3.56329759e-4	-9.69947840e-4
								5.5645514e-5	-1.64709006e-4
									1.32354367e-4
									-1.875841e-5

Приложение Б Пример применения вейвлетного преобразования

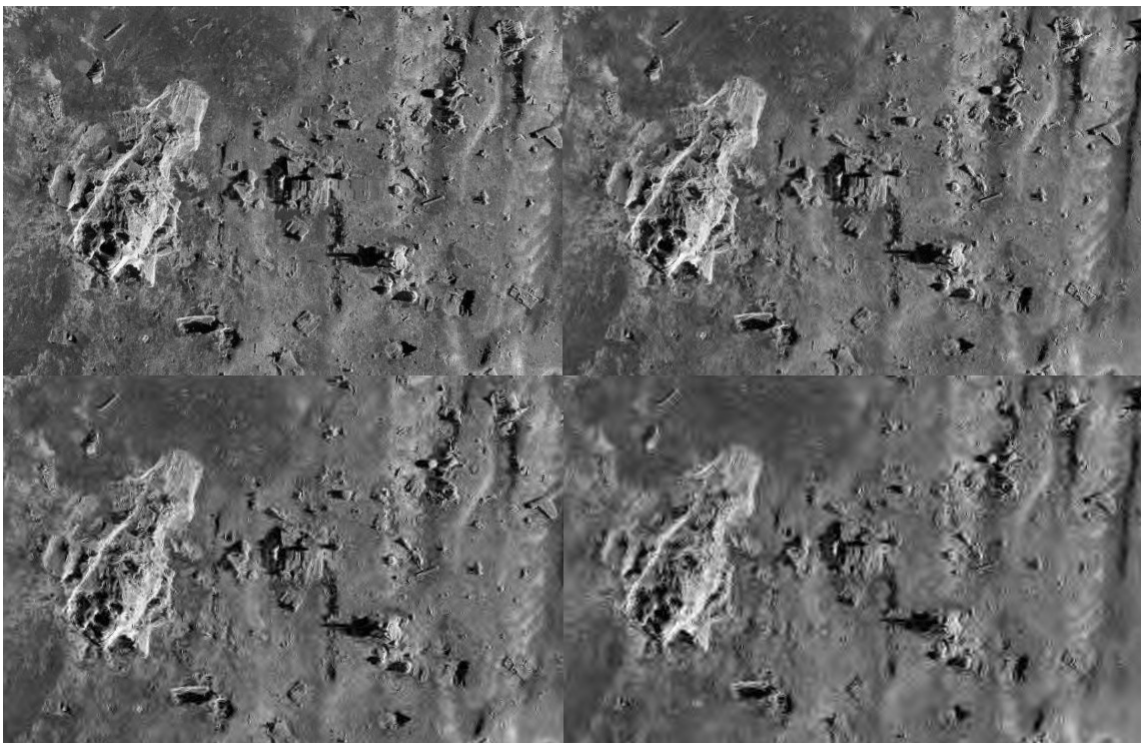

Рисунок Б.1 — Вейвлет-преобразование изображения с сохранением 10% (вверху справа), 5% (внизу слева), 2% (внизу справа) коэффициентов. Вверху слева представлено исходное изображение