Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки Прикладная математика и информатика
Кафедра прикладной математики

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Методы и алгоритмы адаптивного разбиения видеокадров на блоки кодирования в системе видеокодирования H.265/HEVC

УДК 004.932.4

Студент

Группа	ФИО	Подпись	Дата
8БМ41	Тё Дмитрий Юрьевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Марченко Владислав Владимирович	к.т.н.		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Конотопский Владимир Юрьевич	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Анищенко Юлия Владимировна	к.т.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Зав. кафедрой	Гергет Ольга Михайловна	к.т.н.		

Томск – 2016 г.

Содержание

Введение

За последние 30 лет цифровое видео превратилось из нескольких небольших исследовательских проектов в гигантскую индустрию, охватывающую множество областей человеческой деятельности. Объемы данных цифрового видео, передаваемые по различным сетям связи в последние годы удваиваются каждый год [1]. Все это привело к резкой интенсификации исследований, направленных на разработку методов и алгоритмов сжатия цифровых видеоданных. Результатом исследований, проводимых с 2006 года группой JCT-VC, стало принятие в 2013 году нового стандарта видеокодирования H.265/HEVC. Алгоритмы видеосжатия, легшие в основу нового стандарта, потенциально обеспечивают почти двукратное повышение степени сжатия видеоданных по сравнению с теми, что использовались в стандарте предыдущего поколения H.264/AVC. С другой стороны, такое повышение степени сжатия достигается за счет многократного увеличения количества вариантов алгоритмов кодирования видеоизображения в системах нового поколения, что приводит к многократному увеличению вычислительной сложности процедуры видеокодирования. Как показал предварительный анализ вычислительных затрат на кодирование, особенно «узким» местом в системах кодирования H.265/HEVC является выбор варианта разбиения видеоизображения на блоки кодирования, обеспечивающий наибольшую степень сжатия. Все это и обусловило выбор темы магистерской диссертации, ее цели, объекта и предмета исследования.

Целью работы является разработка методов и алгоритмов, обеспечивающих существенное снижение вычислительных затрат при кодировании видеоданных в системах кодирования нового стандарта H.265/HEVC.

Объектом исследования являются методы и алгоритмы видеокодирования, используемые в системах кодирования последнего поколения и удовлетворяющих новому стандарту HEVC.

Предметом исследования является разработка быстрого метода выбора варианта разбиения видеоизображения на блоки кодирования и его алгоритмической реализации.

Актуальность исследования. Интенсивные работы по созданию нового стандарта видеокодирования начались с 2006 г. В 2013 г. по результатам этих исследований был принят новый стандарт – H.265/HEVC. Существенного повышения степени сжатия по сравнению со стандартом предыдущего поколения в HEVC удается добиться за счет адаптивного разбиения видеоизображения на блоки кодирования. Такое разбиение подстраивается в процессе кодирования под характер изображения, обеспечивая максимально высокую степень сжатия. С другой стороны, в новом стандарте не определены критерии выбора варианта разбиения изображения. Выбор такого критерия оставлен разработчикам конкретных кодирующих систем. В простейшем варианте, реализованном разработчиками нового стандарта и опубликованном на сайте JCT-VC, осуществляется перебор всех вариантов разбиения изображения. Для каждого варианта полностью выполняется процедура кодирования. Выбирается вариант, обеспечивающий наилучшую степень сжатия видеоданных при наименьшей степени искажений изображения, вносимых при кодировании. Очевидно, что такой подход оказывается крайне неэффективным с точки зрения объема необходимых для кодирования вычислений. В результате кодирующая система обеспечивает кодирование одного видеокадра высокого разрешения за 2-3 мин. даже на современных высокопроизводительных компьютерных платформах. Такое время кодирования одного видеоизображения ставит под вопрос саму возможность практического использования кодирующих систем, основанных на новом

стандарте (на каждую секунду видео приходится 30 – 60 видеокадров). В области разработки быстрых методов и алгоритмов, обеспечивающих существенное сокращение вычислительных затрат на кодирование в последние три года (после принятия стандарта) ведутся активные исследования [2–7]. Большое внимание в этих исследованиях уделяется процедуре выбора варианта разбиения изображения на блоки кодирования [8–13]. Данная работа является развитием и продолжением этих исследований, что и определяет ее актуальность.

Практическая значимость результатов ВКР. Основные результаты магистерской работы были получены автором в рамках выполнения комплексного проекта «Предоставление услуг мультимедийного вещания в сетях общего пользования Интернет, основанных на технологиях пиринговых сетей и адаптивной передачи потоков данных» выполняемого в рамках Постановления Правительства России от 9 апреля 2010 г. №218 при финансовой поддержке Министертва образования и науки Российской Федерации. Разработанные методы доведены до алгоритмической и программной реализации в виде библиотеки классов на языке C++ и включены в коммерчески распространяемый группой компаний Элекард программный продукт «Elecard Codec SDK». По предварительным оценкам, использование разработанных автором программных модулей позволило более чем в два раза сократить вычислительную сложность процедуры кодирования и, как следствие, поднять скорость кодирования HEVC кодера системы «Elecard Codec SDK». К работе прилагается Акт о внедрении (Приложение В), подтверждающий использование результатов работы в продуктах компании Элекард Девайсез.

Реализация и апробация работы. Основные результаты, полученные при работе над магистерской диссертацией, опубликованы в статье [14] в журнале, входящем в РИНЦ и перечень ВАК. Кроме того, имеется публикация [15] в англоязычном журнале, входящем в базу цитирования

SCOPUS. Получен грант «УМНИК» по теме «Разработка алгоритмов и программного комплекса сжатия цифровых видеопоследовательностей в рамках новейшего стандарта H.265/HEVC», включающей исследования, проводимые в рамках данной магистерской диссертации.

Содержание работы. Диссертация состоит из введения, шести разделов, заключения, трёх приложений. Общий объём 130 страница. Во введении описывается актуальность исследования, ставится цель работы, выделяются объект и предмет исследования, показывается практическая значимость результатов ВКР, кратко описывается содержание работы. В первом разделе данной работы проводится обзор процесса кодирования видеопоследовательностей в соответствии со стандартом H.265/HEVC, вводятся понятия, используемые при обсуждении процесса кодирования, описываются возможности кодирования предоставляемые стандартом. Второй раздел посвящен рассмотрению различных этапов принятия решений, в процессе поиска наилучшего варианта кодирования видео, проводится их анализ, на основании которого принимается решение о необходимости разработки алгоритмов быстрого разбиения кадра на блоки кодирования в системах кодирования H.265/HEVC. В третьем разделе проводится обзор существующих подходов к ускорению процесса кодирования видео, путём сокращения числа перебираемых вариантов разбиений кадра. На основании проведенного анализа разработан новый алгоритм, позволяющий достичь большего ускорения сжатия видеоизображений. Предложены модификации, позволяющие регулировать степень сжатия видеопоследовательностей за счёт изменения вычислительной сложности процесса кодирования. В четвертом разделе описывается программная реализация разработанных алгоритмов в виде библиотеки классов на языке программирования C++ и проводится анализ эффективности полученного решения. Пятый раздел посвящен финансовому менеджменту процесса разработки алгоритмов быстрого разбиения кадра на

блоки кодирования. В шестом разделе проводится анализ производственной безопасности, охраны окружающей среды, защите в чрезвычайных ситуациях и правовым и организационным вопросам обеспечения безопасности, связанным с разработкой и реализацией алгоритмов адаптивного разбиения кадра на блоки кодирования. В заключении подводятся итоги работы и делаются выводы по результатам выполненных работ.

Личный вклад автора. Разработка кодирующей системы стандарта HEVC компании Элекард велась группой исследователей и разработчиков. Автор работы в этой группе отвечал за решение задачи быстрого выбора варианта разбиения изображения на блоки кодирования. Таким образом, основные результаты магистерской диссертации получены лично автором.

1 Обзор кодирования видео в рамках H.265/HEVC

В данном разделе проводится обзор процесса кодирования видеопоследовательностей в соответствии со стандартом H.265/HEVC. Вводятся понятия, используемые при обсуждении процесса кодирования. Описываются возможности, позволяющие достигать высокой степени сжатия изображений, предоставляемые стандартом H.265/HEVC. Показана необходимость в разработке быстрых методов выбора режимов кодирования

1.1 Описание процесса сжатия гибридного кодера H.265/HEVC

H.265 или HEVC (англ. High Efficiency Video Coding – высокоэффективное кодирование видеоизображений) – стандарт сжатия видео, разработанный объединенной командой экспертов видеокодирования JCT-VC, в которую входят коллективы ITU-T Video Coding Experts Group (VCEG) и ISO/IEC Moving Picture Experts Group (MPEG), в 2013 году [16]. Основной целью разработчиков стандарта H.265/HEVC является повышение эффективности алгоритмов сжатия по сравнению с существующими стандартами. Кодирование видео в соответствии с H.265/HEVC позволяет до двух раз повысить степень сжатия при том же уровне визуального качества изображения по сравнению с предшествующим стандартом H.264/AVC [20].

Повышение степени компрессии видеопоследовательностей в рамках H.265/HEVC достигается благодаря введению множества новых методов и алгоритмов обработки видео. Необходимо заметить, что текст стандарта содержит только описание битового потока, получаемого в результате процесса кодирования. Таким образом, выбор набора методов и алгоритмов при кодировании в соответствии с H.265/HEVC остается за разработчиком кодера (кодирующей системы).

H.265/HEVC использует гибридный подход кодирования видео. Т.е. для сжатия используются внутрикадровое (интра) и межкадровое (интер) кодирование потока видеоданных [17].

На рис. 1 представлена упрощенная схема гибридного видео кодера, соответствующего стандарту H.265/HEVC. Алгоритм его работы заключается в следующем [16]. Кодируемое изображение делится на квадратные области, блоки дерева кодирования (англ. coding tree unit, CTU), содержащие блоки кодирования (англ. coding unit, CU), которые, в свою очередь, могут также быть разбиты на блоки предсказания (англ. prediction unit, PU) и блоки преобразования (англ. transform unit, TU). Первый кадр видеопоследовательности, а в дальнейшем и все прочие ключевые кадры, кодируются с использованием только внутрикадрового предсказания (интра-предсказание). Это означает, что для устранения избыточности изображения значения интенсивности цветовых компонент блока PU предсказываются на основании значений интенсивности соседних ему пикселей, принадлежащих этому же изображению и никак не зависящих от других кадров. Ключевые кадры называют I-кадрами (от англ. Intra-coded frame). Для не ключевых кадров видеопоследовательности кроме внутрикадрового доступен также межкадровый режим предсказания (интер-предсказание). Данный режим кодирования заключается в выборе опорного кадра и определении вектора движения (англ. motion vector, MV), которые необходимо передать в алгоритм компенсации движения (англ. motion compensation, MC), для предсказания значений интенсивности цветовых компонент блока PU. Если для кадра в качестве опорных разрешены только кадры, в порядке вывода на экран предшествующие текущему, то он называется P-кадром (от англ. Predicted frame). Если в качестве опорных разрешены кадры с обеих сторон от текущего в порядке вывода на экран, то он называется B-кадром (от англ. Bi-predictive frame).

После предсказания значений пикселей, вычисляется остаточный сигнал *R*:

$$R(x, y) = P_{org}(x, y) - P_{pred}(x, y), \quad (1)$$

где *x*, *y* – координаты пикселя; $P_{org}(x, y)$ и $P_{pred}(x, y)$ – исходное и предсказанное значения пикселей соответственно. Этот сигнал подвергается дискретному косинусному преобразованию (англ. discrete cosine transform, DCT) и квантованию. Полученные значения дополнительно сжимаются энтропийным кодированием и передаются в выходной поток вместе с информацией о предсказании.

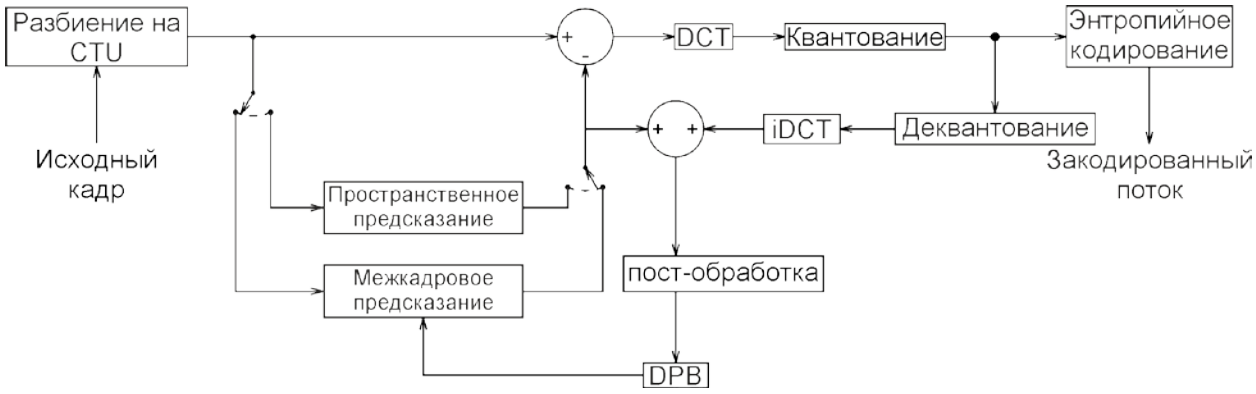


Рис. 1. Упрощенная схема гибридного видео кодера, соответствующего стандарту H.265/HEVC

Для того чтобы в дальнейшем закодированные изображения можно было использовать в качестве опорных кадров интер-предсказания, кодер должен также выполнить задачу декодера – восстановить изображение на основании информации о предсказании и остаточном сигнале. Для этого квантованные коэффициенты деквантуются и подвергаются обратному дискретному косинусному преобразованию (iDCT). В результате получается восстановленный остаточный сигнал R_{rec}. Значения интенсивностей цветовых компонент восстановленного изображения P_{rec} рассчитываются по формуле:

$$P_{rec}(x, y) = P_{pred}(x, y) + R_{rec}(x, y), \quad (2)$$

где x, y – координаты пикселя; $P_{pred}(x, y)$ – предсказанное значение пикселя, $R_{rec}(x, y)$ – восстановленный остаточный сигнал. P_{rec} в дальнейшем может быть подвергнут пост-обработке для устранения последствий квантования и артефактов блочного кодирования. Восстановленный кадр сохраняется в буфер декодированных изображений (англ. decoded picture buffer, DPB) для дальнейшего использования при кодировании последующих кадров. Стоит отметить, что порядок кадров в закодированном потоке может отличаться от порядка кадров исходной видеопоследовательности.

1.2 Описание возможностей кодирования H.265/HEVC

Рассмотрим некоторые особенности кодирования в рамках стандарта H.265/HEVC [16]:

- *Блоки дерева кодирования, CTU*. Как было сказано ранее, при кодировании изображение разбивается на квадратные области. Наибольшей единицей такого разбиения является CTU. В пределах одного кодируемого потока размер CTU является фиксированным. H.265/HEVC поддерживает следующие размеры CTU: 16×16, 32×32 и 64×64 пикселей.
- *Блоки кодирования, CU*. В H.265/HEVC блоки CTU могут быть рекурсивно разбиты на несколько уровней подблоков, называемых блоками кодирования CU, образуя квадродерево – CTU или содержит ровно один CU, совпадающий по размеру с самим CTU, или содержит четыре CU, имеющих длину сторон равную половине длины CTU, которые так же могут быть разбиты аналогичным образом. Очевидно, наибольший размер блока CU ограничивается размером CTU. Наименьший размер CU – 8×8 пикселей. CU, не имеющие подразбиений, будем называть листами квадродерева. Такие CU содержат информацию о том, какой вид кодирования,

пространственный или межкадровый, необходимо использовать для соответствующей ему области изображения. Каждому листовому CU соответствуют определенное разбиение на блоки предсказания PU и дерево блоков преобразования (TU).

– *Блоки предсказания, PU.* Данный вид блоков отвечает за информацию о предсказании значений интенсивностей цветовых компонент. PU может совпадать по размерам с соответствующим ему блоком CU или быть разбит. Все возможные разбиения PU представлены на рис. 2. Разбиения, обозначенные как 2Nx2N и NxN называются квадратными разбиениями, 2NxN и Nx2N выделяются в группу симметричных разбиений (symmetric motion partition, SMP), nLx2N, nRx2N, 2NxnU и 2NxnD – асимметричные разбиения (англ. asymmetric motion partition, AMP). В формировании обозначений разбиений N обозначает половину размера CU; для AMP n – четверть размера CU, буквы L, R, U и D при символе n означают, что меньшим подблоком является «левый», «правый», «верхний» и «нижний» соответственно. В режиме внутрикадрового предсказания доступны только квадратные разбиения. При этом разбиение NxN доступен только для CU наименьшего размера. Для межкадрового предсказания доступны все виды разбиений PU при условии соблюдений следующих ограничения[18]:
    - разбиения NxN доступны только для CU наименьшего размера;
    - запрещены разбиения на PU размером 4×4 пикселя, как следствие для наименьшего размера CU равного 8×8 пикселей разбиение NxN запрещено вовсе;
    - AMP режимы разбиений запрещены для N = 4;

- для режима предсказания Skip (см. подпункт «*Определение вектора движения»*) может использоваться только разбиение 2Nx2N.

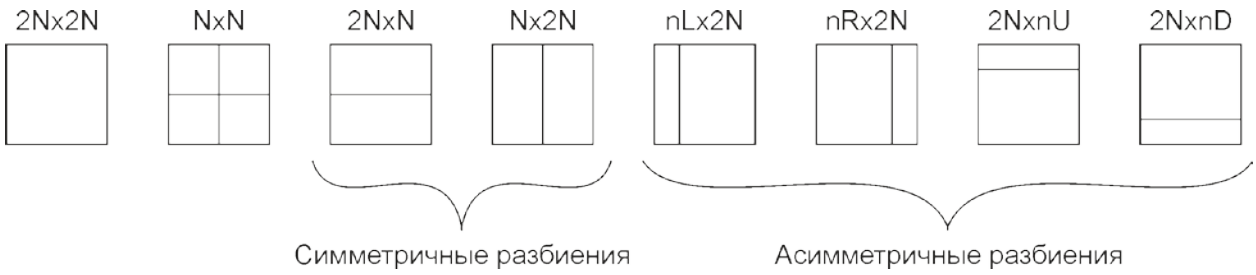


Рис. 2. Режимы разбиения блоков PU

– *Блоки преобразования, TU*. Блоки TU содержат информацию о преобразовании остатков *R* (1) . Размер TU может совпадать с размером соответствующего CU, или он может быть разбит на блоки меньшего размера, образуя квадродерево, корневой уровень которого совпадает с уровнем соответствующего этому дереву CU.
– *Определение вектора движения MV*. В H.265/HEVC используется эффективное предсказание вектора движения (англ. advanced motion vector prediction, AMVP), включающее в себя определение наиболее вероятных кандидатов на основании данных из соседних блоков PU и опорных кадров. Предсказания значений пикселей, основанные только на предсказанных AMVP векторах, называется «слиянием» (англ. merge). Если при этом нет необходимости кодировать остатки *R* (1), в поток записывается сигнализирующий об этом флаг. Такой режим кодирования позволяет достигнуть наибольшей степени сжатия блока CU и носит характерное название – режим Skip (от англ. – пропустить).
– *Компенсация движения, MC*. Вектора MV рассчитываются с четверть-пиксельной точностью. Для интерполяции значений пикселей с дробным вектором движения используются фильтры

седьмого или восьмого порядков. В качестве опорных кадров могут использоваться несколько изображений.

– *Внутрикадровое предсказание*. Для предсказания значений пикселей в областях пространственного предсказания используются восстановленные значения граничных пикселей соседних блоков. В H.265/HEVC поддерживается 35 режимов пространственного предсказания: 33 угловых режима, режим Planar и режим DC. Внутрикадровый режим текущего блока PU может быть унаследован от соседних PU. При этом в выходной поток сохраняется только индекс PU, от которого производится наследование, в однозначно устанавливаемом списке кандидатов. Этот список кандидатов в тексте стандарта [19] носит название MPM (от англ. Most Probable Modes – наиболее вероятные режимы).

1.2 Основные результаты и выводы по разделу

В данном разделе представлен краткий обзор методов и алгоритмов сжатия цифровых видеоданных, вошедших в стандарт H.265/HEVC. На основании обзора можно сделать следующие выводы.

1. H.265/HEVC является стандартом гибридного кодирования, то есть при сжатии видео кадры могут кодироваться во внутрикадровом (пространственном) или межкадровом режиме.
2. Обработка изображения при кодировании осуществляется поблочно. Стандарт допускает большую вариативность при выборе размеров и положения каждого обрабатываемого блока.
3. Каждый блок при кодировании может быть обработан по одному из множества вариантов алгоритмов кодирования. Критерий выбора неилучшего варианта в стандарте не определён, а оставлен на усмотрение разработчиков конкретных систем кодирования.

В целом можно сделать вывод о том, что существенное повышение степени сжатия видеоданных в рамках нового стандарта достигается за счёт расширения списка возможных методов и алгоритмов обработки видеоизображений. Оборотной стороной такого расширения является существенное повышение вычислительной сложности процедуры видеокодирования. Это делает особенно актуальными исследования, направленные на разработку быстрых методов и алгоритмов видеокодирования в рамках нового стандарта H.265/HEVC.

2 Процедура принятия решений в программной реализации кодирующей системы

В данном разделе подробно рассматриваются различные этапы принятия решений, в процессе поиска наилучшего варианта кодирования видео. В результате исследования этой процедуры в качестве перспективного, с точки зрения ускорения процесса сжатия видео, выбран этап разбиения кадра на блоки кодирования CU и блоки предсказания PU.

2.1 Критерий сравнения оптимальности вариантов кодирования

Определим критерий, по которому можно сравнивать варианты кодирования. В качестве такого критерия традиционно используется метрика, называемая в англоязычной литературе *RDC* (от англ. Rate-Distortion Cost) [21]. Величина *RDC* складывается из двух слагаемых:

$$RDC = D + \lambda \cdot R, \tag{3}$$

где $D = \sum_{x,y} \left| P_{org}(x,y) - P_{rec}(x,y) \right|$, $x, y = 0, 1, \ldots, N-1$; $P_{org}(x,y)$ – значения отсчетов кодируемого блока; $P_{rec}(x,y)$ – значения декодированных отсчетов; N – размер кодируемого блока в пикселях; R – количество бит, представляющих кодируемых блок в битовом потоке на выходе энтропийного кодера; λ – множитель Лагранжа. Величина *RDC*, таким образом, определяется, с одной стороны, величиной искажений *D*, вносимых в изображение при кодировании на этапе квантования спектральных коэффициентов, с другой стороны – степенью сжатия кодируемого блока *R*.

Заметим, что для вычисления величины *R* для определенного варианта кодирования, необходимо полностью провести процесс кодирования этого варианта. Для получения же величины *D* необходимы значения $P_{rec}(x,y)$, которые получаются в результате декодирования. Следовательно, для расчёта

величины *RDC*, соответствующего одному варианту кодирования, необходимо выполнить полный цикл кодирования и декодирования участка изображения. Таким образом, вычислительная сложность оценки стоимости одного варианта кодирования велика.

Принятие решения о выборе того или иного варианта кодирования, будь то разбиение блоков кодирования, режим внутрикадрового или межкадрового предсказания, осуществляется так, чтобы минимизировать величину *RDC*. Такой подход обеспечивает максимальную степень сжатия видеоизображения при минимизации искажений, вносимых в изображение при кодировании. Подробнее задача оптимизации кодирования видеокадров описывается в разделе 2.5.

2.2 Выбор структуры разбиений блоков

Блоки CU, PU и TU в H.265/HEVC могут иметь различный размер (см. раздел 1). Выбираемый при кодировании размер блоков может зависеть от различных факторов. В частности, при внутрикадровом предсказании, чем сложнее текстура в рассматриваемой области изображения, тем, вероятно, меньший размер блоков следует использовать. Для межкадрового кодирования такое не верно. При удачном поиске вектора движения и выборе опорного кадра предсказание может давать малое количество искажений *D* для больших блоков PU, следовательно, нет необходимости кодировать множество дополнительных структур, а значит и величина *R* будет меньше.

Для поиска оптимального решения о разбиении CTU на подблоки необходимо вычислить *RDC* для всех возможных комбинаций разбиений каждого блока, что и приводит к крайне высоким вычислительным затратам. Так, например, для каждого блока изображения размером 64×64 пикселя, величина *RDC* вычисляется 1 раз для всего блока целиком, 4 раза – для четырех поблоков размером 32×32, 16 раз – для подблоков размером 16×16,

64 раза – для подблоков размером 8×8 и 256 раз – для подблоков размером 4×4.

Пусть CU_c – блок CU, по которому принимается решение о разбиении; CU_{sub1}, CU_{sub2}, CU_{sub3}, CU_{sub4} – 4 непосредственных подблока CU_c; $RDC(CU)$ – условное обозначение функции, вычисляющей значение *RDC* блока *CU*. Тогда, для принятия решения о необходимости разбиения CU_c выполняется следующая процедура[22]:

Процедура *NeedSplit*(*CU*):

1. Вычислить $RDC_{CU} = RDC(CU)$
2. Если размер *CU* равен минимальному размеру CU
   2.1. Принять решение о не разбиении
   2.2. Вернуть значение RDC_{CU} и закончить процедуру
3. [CU_{sub1}, CU_{sub2}, CU_{sub3}, CU_{sub4}] = подблоки *CU*
4. Вычислить $RDC_{subCU1} = NeedSplit(CU_{sub1})$
5. Вычислить $RDC_{subCU2} = NeedSplit(CU_{sub2})$
6. Вычислить $RDC_{subCU3} = NeedSplit(CU_{sub3})$
7. Вычислить $RDC_{subCU4} = NeedSplit(CU_{sub4})$

8.1. Если $RDC_{CU} < RDC_{subCU1} + RDC_{subCU2} + RDC_{subCU3} + RDC_{subCU4}$
   8.1.1. Принять решение о не разбиении
   8.1.2. Вернуть значение RDC_{CU} и закончить процедуру

8.2. Иначе
   8.2.1. Принять решение о разбиении
   8.2.2. Вернуть значение $RDC_{subCU1} + RDC_{subCU2} + RDC_{subCU3} + RDC_{subCU4}$ и закончить процедуру

В процессе кодирования кадра разбиение изображения на блоки является начальной точкой для всех дальнейших вычислений. Это означает, что при отсечении каждого варианта разбиения, так же отсекаются, во-

первых, все дальнейшие подразбиения, а во-вторых, все дополнительные вычисления, описанные в подразделах 2.3–2.5.

Таким образом, разработав методы и алгоритмы, позволяющие определять наиболее близкое к оптимальному разбиение кадра на блоки кодирования и блоки предсказания, потенциально можно добиться значительного прироста в скорости кодирования. Разработке и исследованию таких методов и алгоритмов посвящена данная работа.

2.3 Выбор режима предсказания

Как было показано в разделе 2.2, количество проверяемых разбиений велико, вследствие чего приходится много раз выполнять сложные в вычислительном смысле расчёты значения *RDC*. Количество вычислений величины *RDC* не ограничивается числом возможных разбиений. Для каждого PU необходимо выбирать режим предсказания. Этот выбор так же совершается на основании оценки *RDC*. Число режимов предсказаний зависит от типа предсказания – внутрикадровое (пространственное) или межкадровое.

2.3.1 Выбор режима пространственного предсказания

При пространственном предсказании для формирования предсказанного сигнала используются восстановленные ранее значения пикселей, принадлежащих соседним блокам PU, граничащим с текущим блоком (рис. 3). Значения этих пикселей используются для предсказания в соответствии с одним из 35 доступных алгоритмов, называемых режимами предсказаний. Каждый режим имеет свой определенный порядковый номер – индекс. Все режимы можно условно разделить на три группы:

– *Planar* (режим 0) вычисляет предсказанные значения пикселей блока предсказания как среднее арифметическое полученных в результате линейной интерполяции в горизонтальном и вертикальном направлениях величин.
– *DC* (режим 1) применяется для участков изображения с монотонной текстурой. Блок PU, предсказанный в режиме DC присваивает всем пикселям одинаковое значение, равное среднему арифметическому значений пикселей соседей.
– Оставшиеся 33 режима предсказания (2 – 34) называются *угловыми* режимами. При угловом предсказании значения пикселей соседей распространяются в направлении, соответствующем выбранному режиму (рис. 4).

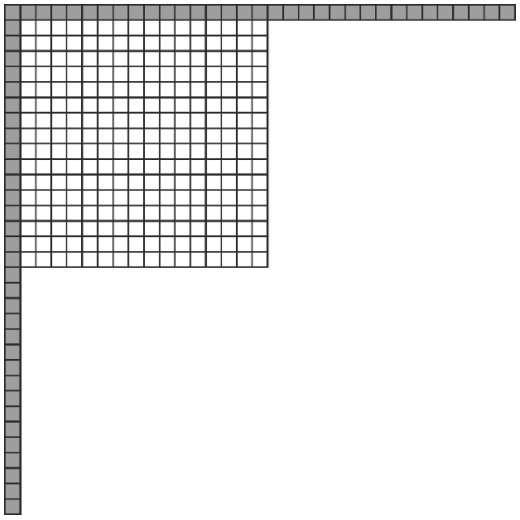


Рис. 3. Пиксели, используемые для пространственного предсказания (серые) значений пикселей текущего PU (белые)

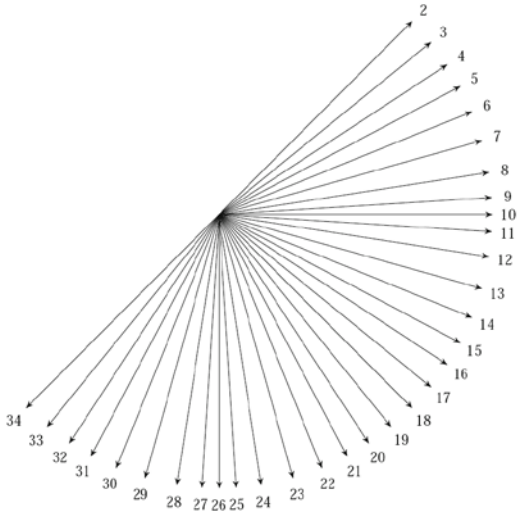


Рис. 4. Соответствие направлений пространственных предсказаний с их индексами.

Таким образом, для поиска оптимального режима пространственного предсказания необходимо выполнить 35 оценок значения *RDC*. На практике этот поиск осуществляется по менее точному, но не требующему проведения

полного цикла кодирования-декодирования для каждого режима алгоритму RMD (от англ. Rough Mode Decision – грубая оценка режимов) [22]. Вводится дополнительная метрика RDC_{SATD} (от англ. Sum of Absolute Hadamard Transformed Differences):

$$RDC_{SATD} = SATD + \lambda \cdot R_{pred}, \quad (4)$$

$$SATD = (\sum_{i,j} |DiffT(i, j)|) / 2, \quad (5)$$

$$DiffT = had(P_{org} - P_{pred}), \quad (6)$$

где P_{org} – исходное изображение; P_{pred} – предсказанное изображение; $had(.)$ – преобразование Адамара; R_{pred} – количество бит, требуемое для кодирования информации о предсказании. Как видно, для расчёта этой метрики нет необходимости выполнять полное кодирование блока, так что оценка стоимости кодирования блока по данному критерию значительно быстрее оценки по метрике *RDC*.

По метрике RDC_{SATD} выбирается фиксированное число лучших кандидатов, для которых выполняется полная проверка по критерию *RDC*.

Таким образом, для выбора режима пространственного предсказания можно уйти от трудоёмкой проверки всех 35 вариантов, предварительно отсеяв большую часть режимов на основании упрощенной оценки RDC_{SATD}.

2.3.2 Выбор режима межкадрового предсказания

Межкадровое кодирование, в отличие от пространственного, использует более сложные методы предсказаний. Во-первых, гораздо больше вариантов разбиения CU на PU (см. раздел 1). Во-вторых, для каждого блока PU необходимо решить, какой вид межкадрового предсказания использовать: однонаправленный или двунаправленный. В-третьих, необходимо выбрать один опорный кадр в случае однонаправленного предсказания, или два опорных кадра в двунаправленном случае (по одному в каждом

направлении), из списка доступных опорных кадров. Наконец, необходимо выполнить поиск наилучших векторов движения MV текущего блока PU относительно опорных кадров. При этом вектора MV определяются с четверть-пиксельной точностью.

Если при пространственном кодировании большая часть разбиений на блоки – это разбиение CTU на CU (кроме CU нижнего уровня, которые ещё один раз могут быть разбиты на 4 PU), то в межкадровом режиме CU на каждом уровне может быть разбит восемью разными способами на PU. Ограничения на разбиения описаны в разделе 1. Во избежание проверки наименее вероятных асимметричных AMP режимов используется следующее правило[18]:

- В первую очередь проверяются режимы 2Nx2N, 2NxN и Nx2N.
- Если лучшим является горизонтальный SMP режим 2NxN, то далее проверяются только горизонтальные AMP режимы 2NxnU и 2NxnD.
- Если лучшим является вертикальный SMP режим Nx2N, то далее проверяются только вертикальные AMP режимы nLx2N и nRx2N.
- Если лучшим является квадратный режим 2Nx2N, то проверяются все AMP режимы.

Таким образом, вертикальные AMP разбиения не проверяются в случае, если горизонтальное SMP разбиение было признано лучшим, среди SMP режимов и варианта без разбиения блоков; горизонтальные AMP разбиения не проверяются в случае, если лучшим было признано вертикальное SMP разбиение.

В процессе кодирования восстановленные кадры попадают в буфер декодированных изображений DPB. Для межкадрового предсказания могут быть использованы только кадры, хранящиеся в этом буфере. Причём для каждого P или B кадра дополнительно формируются списки опорных изображений. Для P кадра такой список один. Он включает в себя только те кадры, которые располагаются перед кодируемым кадром в порядке вывода

на экран. Для B кадров используется также второй список, содержащий кадры, расположенные в таком порядке после кодируемого кадра. Блок предсказания PU может ссылаться только на кадры, имеющиеся в соответствующих ему списках опорных кадров.

Для применения техники компенсации движения, необходимо правильно найти вектор MV. Для этого используются алгоритмы оценки движения ME (от англ. Motion Estimation). Опустим описание возможных алгоритмов ME. Логично предположить, что соседние блоки предсказания с высокой долей вероятности будут иметь схожие вектора движения относительно одних и тех же опорных кадров. В H.265/HEVC используются так называемые списки кандидатов для слияния движения, основанные на этом предположении. При кодировании блок PU может унаследовать информацию об опорном кадре и векторе движения у ранее закодированных соседних блоков PU. В дальнейшем алгоритмы ME могут быть запущены с начальной точкой, сдвинутой на предсказанный вектор. Тогда, даже если оптимальный вектор MV не точно совпадает с унаследованным от соседнего блока вектором, в закодированный поток необходимо отправить только информацию о разности между предсказанным и оптимальным векторами. В тексте стандарта эта разница называется MVD (от англ. Motion Vector Difference).

Несмотря на то, что оптимальный вектор MV может не совпадать с предсказанным вектором из списка кандидатов для слияния, для тех случаев, когда они всё же совпадают, существуют отдельные режимы кодирования – Merge (англ. «слияние») и Skip (англ. «пропуск»). Для блока PU могут быть установлены флаги, сигнализирующие об использовании этих режимов. Режим Merge может быть использован сам по себе для любых размеров PU, в то время как Skip используется только совместно с Merge и только для разбиений 2Nx2N. Применение режима Merge подразумевает кодирования в поток только номера наследуемого кандидата и информации об остаточном

сигнале. В режиме Skip кодируется только номер кандидата, информация об остатках «пропускается», отсюда и название режима. Другими словами режим Merge подразумевает объединение движения текущего блока PU с одним из соседних блоков, а режим Skip кроме этого означает, что движение подобрано «идеально». Под «идеальным» подразумевается то, что уровень шумов D_{pred}, возникающий при предсказании, очень мал, а остаточный сигнал таков, что полученными в результате выполнения DCT и квантования коэффициентами можно пренебречь. В любом случае, это решение принимается на основании оценки *RDC*.

На этапе межкадрового предсказания большу́ю часть вычислений можно отсеять, оптимизировав последовательность проверяемых разбиений на PU и проверки режимов Merge и Skip. В работе [18] описываются исследования эффектов от применения различных подходов к определению режима межкадрового кодирования. В некоторых случаях авторам удаётся сократить время кодирования на 60% при незначительном ухудшении степени сжатия. По большей части такое ускорение достигается за счёт отсева большого количества проверок не Merge и не Skip режимов, для оценки которых необходимо выполнять трудоёмкий процесс оценки движения.

Таким образом, для ускорения процесса кодирования при выполнении межкадрового предсказания имеет смысл научиться быстро определять оптимальный режим разбиения блока кодирования CU на блоки предсказания PU и оптимальный режим кодирования этих PU, который может так же включать в себя поиск оптимального вектора движения.

2.4 Выбор размера блока преобразования TU

Имея предсказанные значения интенсивностей пикселей, можно сформировать остаточный сигнал *R* по формуле (1). Остаточный сигнал *R*

далее подвергается дискретному косинусному преобразованию DCT и дальнейшему квантованию. При этом, область, для которой выполняется DCT и квантизация, так же может быть разбита на квадродерево, корнем которого является соответствующий данной области блок кодирования CU. Выбор оптимального разбиения CU на TU, аналогично разбиению CU на PU, основывается на метрике *RDC* (3).

Стоит напомнить, что наибольший размер блока кодирования CU, допустимый в H.265/HEVC – 64×64 пикселя, а наибольше размер TU – 32×32 пикселя. Таким образом, если необходимо выполнить DCT для области изображения, соответствующей CU размера 64×64 пикселя, соответствующее дерево TU всегда разбивается хотя бы на один уровень.

Стоит отметить, что при кодировании блока CU в режиме пространственного предсказания блоки TU соответствуют областям предсказаний. Режим пространственного предсказания определяется один на весь блок CU, но предсказание делается для каждого TU отдельно. Блок CU самого нижнего уровня может быть также разбит на четыре PU в режиме NxN (см. раздел 1). В таком случае каждый из четырех PU имеет свой отдельный режим предсказания, но и в этом случае области предсказаний совпадают с разбиением на блоки TU.

Как говорилось в разделе 2.2, количество проверяемых разбиений CTU на CU велико. Дерево блоков TU может иметь такую же глубину, как дерево CU, соответственно при выборе оптимального размера блока преобразования так же выполняется большое количество вычислений. Фактически, эти вычисления являются наиболее трудоёмкими во всём процессе кодирования, так как включают в себя прямое и обратное косинусное преобразования, квантование и деквантование. Вдобавок, выбор разбиения на TU производится для каждого проверяемого CU.

Учитывая, что поиск оптимального разбиения области изображения на блоки преобразования включает в себя большое количество трудоёмких

вычислений, перспективным является проведение исследований для дальнейшей разработки методов и алгоритмов, позволяющих ускорить процесс сжатия видео на этом этапе.

2.5 Оптимизация битовой скорости и искажений (RDO)

Пусть каждый CTU кодируется в определённом режиме. Этот режим включает в себя используемый режим предсказания с соответствующими ему параметрами и информацию о дереве блоков преобразований. Тогда задача кодера – решить, каким именно образом закодировать видео (какой режим предсказания применить к каждому CTU на каждом кадре последовательности), чтобы выходной поток при этом был оптимальным с точки зрения битовой скорости R и визуальных искажений D, вносимых выбранным режимом кодирования. В видеокодировании такая задача решается методом RDO (от англ. Rate-Distortion Optimization – оптимизация битовой скорости и искажений) [8].

Пусть изображение разбито на N блоков CTU, множество всех таких блоков обозначим $P = \{\mathrm{CTU}_n\}$, $n = 0, \ldots, N - 1$; режим кодирования блока CTU_n обозначим индексом M_n; множество режимов кодирования обозначим $M = \{M_n\}$, n = 0, …, $N - 1$. Тогда задача оптимизации для кодера заключается в поиске наилучшего режима кодирования M_{opt} такого, что для заданного ограничения битовой скорости R_c количество искажений минимально [24]:

$$\min_M(D(P,M)), \text{ при } R(P,M) < R_c . \quad (7)$$

Используя весовой множитель Лагранжа λ можно уйти от ограничения, тогда:

$$M_{opt} = \arg\min_M(RDC(P,M \mid \lambda)), \quad (8)$$

где RDC определяется согласно формуле (1):

$$RDC(P,M \mid \lambda) = D(P,M) + \lambda \cdot R(P,M) . \quad (9)$$

При кодировании изображений предполагается, что метрика *RDC* обладает свойством аддитивности. Тогда, поиск минимума *RDC* для целого кадра сводится к поиску минимумов каждого отдельного блока CTU:

$$\min_{M} \sum_{n=0}^{N-1} RDC(CTB_n, M \mid \lambda) = \sum_{n=0}^{N-1} \min(RDC(CTB_n, M \mid \lambda)). \quad (10)$$

Стоит отметить, что такое предположение не учитывает возможное влияние принимаемого решения для определенного CTU на последующие блоки, так же, как и влияние на будущие кадры, кодируемые в межкадровом режиме.

Используемое значение коэффициента Лагранжа λ зависит от применяемой функции оценки искажений *D* и конфигурации кодирующей системы, и вычисляется непосредственно кодером.

2.6 Основные результаты и выводы по разделу

Проведен анализ основных этапов принятия решений о режимах кодирования видео. На каждом этапе выделены трудоёмкие, с вычислительной точки зрения, места. Показано, что разработка алгоритмов и методов, позволяющих быстро определять наиболее близкое к оптимальному разбиение кадра на блоки кодирования и блоки предсказания, является значимой. Проведенный анализ показал следующее.

1. При принятии решения о выборе оптимального варианта кодирования используется оптимизация битовой скорости и искажений RDO. Эта процедура заключается в поиске режима кодирования, при котором значение метрики *RDC* для каждого отдельного блока кодирования будет минимальной. Для подсчёта значения *RDC* для каждого проверяемого варианта необходимо выполнить полное кодирование и декодирование блока, что является сложным, с вычислительной точки зрения, процессом.

2. При определении оптимального варианта разбиения CTU на CU необходимо выбрать оптимальные режимы для большого количества блоков различных размеров. Оценка стоимости кодирования каждого отдельного блока включает в себя множество вычислений метрики *RDC* на каждом этапе: начиная от выбора режима предсказания и заканчивая принятием решения о разбиении на дерево блоков преобразования TU. Вследствие этого, быстрое принятие решения о проверке или не проверке какого-либо варианта разбиения блока CU может привести к большой экономии времени в процессе кодирования видео.
3. Одним из самых трудоёмких, с вычислительной точки зрения, является этап выбора режима предсказания. Для кадров, кодируемых в режиме пространственного предсказания, это выбор одного из 33 угловых режимов или режимов Planar и DC. Для сокращения вычислений в этом случае применяется метод грубого принятия решения RMD, заключающийся в предварительном отсеве заведомо неоптимальных режимов, с использованием метрики RDC_{SATD}, не требующей вычисления фактических величин искажений *D* и битовой скорости *R*. Для межкадрового предсказания выполняется поиск оптимального разбиения CU на один из восьми доступных режимов PU. Для каждого из PU выполняется трудоёмкая процедура оценки движения. Наиболее перспективными для ускорения местами на этом этапе кодирования являются выбор режима пространственного предсказания и выбор разбиения CU на PU для межкадрового предсказания, который может отсечь множество излишних процедур оценки движения.
4. Разбиение блока кодирования CU на блоки преобразований TU может являться трудоёмкой, с вычислительной точки зрения, задачей, поскольку сверху размер TU ограничивается размером

соответствующего ему CU, а глубина разбиения может достигать четырех на больших размерах CU. Оценка *RDC* каждого блока TU включает в себя трудоёмкие процедуры прямого и обратного дискретного преобразования. Всё это делает процесс определения оптимального разбиения TU одним из потенциальных мест, модификация которого может привести к значительному ускорению процесса кодирования видео в целом.

3 Разработка алгоритма быстрого разбиения кадра на блоки кодирования

Разработке быстрого критерия для выбора размера блока пространственного предсказания, позволившего бы существенно снизить вычислительные затраты на кодирование при незначительном увеличении значения *RDC*, посвящено множество исследований во всем мире. Результаты этих исследований опубликованы в работах [11–16]. В данном разделе на основе анализа опубликованных результатов выбран наилучший по соотношению вычислительной сложности и проигрыша по значению *RDC*. Предложена модификация критерия, обеспечивающая дополнительное сокращение вычислительных затрат при кодировании. Предложен способ регулирования соотношения снижения степени сжатия и сокращения затрачиваемого на кодирование времени.

3.1 Критерии сравнения эффективности алгоритмов сжатия

Для поиска оптимального разбиения кадра на блоки кодирования в процессе сжатия видео необходимо перебирать все возможные варианты, что является трудоёмкой задачей. Ускорение данного процесса подразумевает отсечение проверок некоторых вариантов кодирования или использование какого-либо упрощенного алгоритма оценки стоимости режима кодирования. При таком подходе некоторые оптимальные режимы могут быть отсечены или неправильно оценены как неоптимальные, что приводит к повышению битовой скорости R и искажений D закодированного потока. Различные методы, позволяющие ускорить процесс сжатия видеопоследовательностей, могут производить различный эффект, как с точки зрения битовой скорости и искажений, так и с точки зрения сокращения затрачиваемого на кодирование времени. В связи с этим, возникает необходимость в

определении критериев сравнения эффективности различных подходов к сокращению вычислительной сложности процесса кодирования.

Для сравнения качества различных алгоритмов сжатия цифровых видеоданных традиционно используется метрика Бьёнтегарда [25]. Величина BD-rate (от англ. Bjontegaard delta rate) является оценкой средней относительной разницы битовых скоростей (количество бит в секунду), получаемых при кодировании видеоданных с использованием двух сравниваемых алгоритмов. Положительное значение BD-rate означает рост битовой скорости и, как следствие, снижение на ту же величину степени сжатия видеоданных.

Время кодирования ΔT, которое тот или иной алгоритм позволяет сократить, считается в процентах по следующей формуле:

$$\Delta T = \frac{T_{org} - T}{T_{org}} \cdot 100\%, \quad (11)$$

где – сохраненное время в процентах; T_{org} – время, затрачиваемое на кодирование видеопоследовательности при использовании базового критерия; T – время, затрачиваемое на кодирование видеопоследовательности при использовании альтернативного критерия.

В рамках данной работы время кодирования и качество сжатия сравнивается с результатами работы справочной реализацией кодера HM, если не указано иное. Таким образом, наилучшим методом принятия решений о разбиении блоков кодирования на подблоки будет считаться такой, при котором обеспечивается минимальное значение величины BD-rate при максимальном значении ΔT. Другими словами, такой метод позволяет добиться высокой степени сжатия при низких вычислительных затратах на кодирование.

3.2 Обзор критериев выбора варианта разбиения блоков пространственного предсказания

Разбиения блоков кодирования на подблоки при кодировании осуществляется так, чтобы минимизировать величину *RDC*. Такой подход обеспечивает максимальную степень сжатия видеоизображения в каждом блоке при минимизации искажений, вносимых в изображение при кодировании. С другой стороны, при таком подходе реализуется перебор всех возможных комбинаций разбиений каждого блока, что приводит к крайне высоким вычислительным затратам.

Результаты исследований, направленных на разработку альтернативного критерия для выбора размера блока пространственного предсказания, опубликованы в работах [8–13]. В [8] авторы предлагают останавливать процесс разбиения блока предсказания на подблоки, если для текущего блока величина *RDC* не превышает порога *T*. В статье показано, что степень сжатия видеоданных линейно падает в зависимости от количества блоков, которые должны были быть разбиты, но разбиты не были, при том, что время кодирования сокращается значительно быстрее. Авторами было решено держать показатель ложных прерываний разбиения на уровне 5%, что позволило эмпирически определить зависимость величины порога *T* от размера блока предсказания *N* и параметра квантования *Qp* [19], определяющего шаг квантования спектральных отсчетов остаточного сигнала, в виде

$$T_{64} = 962{,}7 \cdot e^{0{,}126 \cdot Qp}, \quad (12.а)$$

$$T_{32} = 164{,}6 \cdot e^{0{,}148 \cdot Qp}, \quad (12.б)$$

$$T_{16} = 19{,}75 \cdot e^{0{,}187 \cdot Qp}, \quad (12.в)$$

$$T_{8} = 1{,}054 \cdot e^{0{,}254 \cdot Qp}, \quad (12.г)$$

где T_{64}, T_{32}, $T_{16,}$ T_8 – пороговые значения для блоков с размером N = 64, 32, 16, 8 соответственно.

Аналогичный подход изложен в [9]. Он основывается на упрощенной оценке величины *RDC*, обозначаемой в статье *LRDC* (от англ. Low complexity Rate-Distortion Cost), и введении эмпирически подобранных пороговых значений *T*. Если величина *LRDC* меньше уровня порога *T*, то разбиения блока предсказания на подблоки не производится. В противном случае блок разбивается на подблоки. Значение *LRDC* вычисляется как сумма двух слагаемых:

$$LRDC = \sum_{i,j} | A_{i,j} | + \lambda_m \cdot B_m, i, j = 0,1,..., N-1, \quad (13)$$

где величины $A_{i,j}$ представляют результат дискретного двумерного преобразования Адамара остаточного сигнала, полученного в результате вычитания предсказанных значений из отсчетов кодируемого изображения; N – размер блока предсказания в пикселях; B_m – количество бит, требуемых для описания способа пространственного предсказания блока в битовом потоке на выходе энтропийного кодера; λ_m – множитель Лагранжа. Пороговые значения T определяются размером блока предсказания N и параметром квантования *Qp*.

В [10] авторы отмечают высокую пространственную корреляцию глубины разбиения блоков пространственного предсказания. Это позволило им проводить оценку глубины разбиения текущего блока по значению d_p, равному взвешенной сумме глубин разбиения соседних ранее закодированных блоков:

$$d_p = \sum_{i=0}^{3} \alpha_i \cdot d_i, \quad (14)$$

где d_p – оценка глубины разбиения текущего блока; d_i– глубина разбиения четырех соседних с текущим блоков; α_i– весовые коэффициенты: α_0 = 0,3; α_1 = 0,2; α_2 = 0,3; α_3 = 0,2. На рисунке 1 иллюстрируется положение текущего блока (выделен серым) и нумерация соседних блоков.

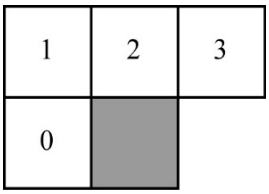


Рисунок 5 – Положение и нумерация соседних блоков

В зависимости от величины значения d_p кодируемый блок относят к одной из четырех групп, для каждой из которых задан возможный диапазон глубин разбиений. Решение о глубине разбиения блока в рамках каждого диапазона принимается на основании вычисления значения *RDC*. Диапазоны значений d_p и соответствующие им диапазоны глубин разбиений приведены в таблице 1.

Таблица 1 – Диапазоны значений d_p и глубины разбиений

Интервал значений d_p	Диапазон глубины разбиений
$d_p \le 0{,}5$	0–1
$0{,}5 < d_p \le 1{,}5$	0–2
$1{,}5 < d_p \le 2{,}5$	1–3
$2{,}5 < d_p$	2–3

В работах [11, 12] в качестве критерия при выборе размера блока пространственного предсказания используются различные меры однородности изображения внутри кодируемого блока. В [11] алгоритм строится «снизу вверх», то есть четыре соседних блока минимального размера $N = 4$ объединяются в один блок с размером $N = 8$, если по крайней мере три из объединяемых блоков признаны однородными. По тому же принципу производится объединение блоков большего размера. Для оценки степени однородности используются средние значения модулей производных

интенсивности изображения по четырем направлениям, рассчитываемые по следующим формулам:

$$g_0 = \sum_{k=0}^{\frac{N}{4}-1}\sum_{l=0}^{\frac{N}{4}-1}\sum_{i=0}^{1}\sum_{j=0}^{\frac{N}{4}-1} | I_{4k+i,4l+2j} - I_{4k+i+2,4l+2j} |, \quad (15.а)$$

$$g_{90} = \sum_{k=0}^{\frac{N}{4}-1}\sum_{l=0}^{\frac{N}{4}-1}\sum_{i=0}^{1}\sum_{j=0}^{\frac{N}{4}-1} | I_{4k+2i,4l+j} - I_{4k+2i,4l+2j+2} |, \quad (15.б)$$

$$g_{45} = \sum_{k=0}^{\frac{N}{4}-1}\sum_{l=0}^{\frac{N}{4}-1}\sum_{i=0}^{1}\sum_{j=0}^{\frac{N}{4}-1} | I_{4k+i,4l+j} - I_{4k+2i+2,4l+2j+2} |, \quad (15.в)$$

$$g_{135} = \sum_{k=0}^{\frac{N}{4}-1}\sum_{l=0}^{\frac{N}{4}-1}\sum_{i=0}^{1}\sum_{j=0}^{\frac{N}{4}-1} | I_{4k+2i+2,4l+j} - I_{4k+2i,4l+2j+2} |, \quad (15.г)$$

где $I_{x,y}$ – интенсивность отсчетов изображения кодируемого блока; N – размер блока, для которого производится расчет. За меру однородности блока принимается величина:

$$C = | g_{\min} - g_{ort} |, \quad (16)$$

где $g_{\min} = \min\{g_0, g_{45}, g_{90}, g_{135}\}$, а g_{ort} – значение производной в ортогональном к направлению g_{min} направлении. Блок считается однородным, если величина C меньше величины порога T, определяемой размером блока N и параметром квантования Qp:

$$T = Qp \cdot N. \quad (17)$$

В качестве мер однородности изображения внутри кодируемого блока в [12] предлагается использовать восемь величин, каждая из которых в работе называется глобальной или локальной сложностью изображения. Четыре значения глобальной сложности, каждое для своего направления, рассчитываются по формулам:

$$G_0 = \sum_{i=0}^{N-1}\sum_{j=0}^{\frac{N}{2}-1} | I_{i,j} - \bar{I} | - \sum_{i=0}^{N-1}\sum_{j=\frac{N}{2}}^{N-1} | I_{i,j} - \bar{I} |, \quad (18.а)$$

$$G_{90} = \sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{N-1} | I_{i,j} - \bar{I} | - \sum_{i=\frac{N}{2}}^{N-1}\sum_{j=0}^{N-1} | I_{i,j} - \bar{I} |, \quad (18.б)$$

$$G_{45} = \sum_{i=0}^{N-1}\sum_{j=0}^{i} | I_{i,j} - \bar{I} | - \sum_{i=0}^{N-1}\sum_{j=i}^{N-1} | I_{i,j} - \bar{I} |, \quad (18.в)$$

$$G_{135} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-i-1} | I_{i,j} - \bar{I} | - \sum_{i=0}^{N-1}\sum_{j=N-i-1}^{N-1} | I_{i,j} - \bar{I} |, \quad (18.г)$$

где $I_{x,y}$ – интенсивность отсчетов изображения кодируемого блока; N – размер блока, для которого производится расчет; $\bar{I}$ – средняя интенсивность изображения в кодируемом блоке. Значения локальной сложности по каждому направлению вычисляются по формуле:

$$L_{ang} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | I_{i,j}^{ang} - \bar{I}^{ang} |, \quad (19)$$

где ang = 0, 90, 45, 135; $I_{i,j}^{\text{ang}}$ – значения производной интенсивности изображения в направлении, заданном значением ang:

$$I_{i,j}^{0} = I_{i-1,j} - I_{i+1,j}, I_{i,j}^{90} = I_{i,j-1} - I_{i,j+1}, \; I_{i,j}^{45} = I_{i-1,j-1} - I_{i+1,j+1}, \; I_{i,j}^{135} = I_{i+1,j-1} - I_{i-1,j+1}; \quad (20)$$

$\bar{I}_{i,j}^{\text{ang}}$ – среднее значение производной интенсивности изображения в заданном направлении.

Авторами [12] предусмотрены три возможных варианта действий при принятии решения о разбиении блока:

1) блок не следует разбивать (тогда дальнейший поиск разбиения прекращается);
2) блок следует разбить (тогда для текущей глубины никаких дополнительных вычислений не производится);
3) точно определиться с разбиением невозможно (тогда разбиение производится по величине *RDC*).

Случай 1 возникает, когда глобальные и локальные сложности границ в каком-либо из четырех направлений меньше заданных пороговых значений для текущего CU и всех его подблоков. Случай 2 возникает, когда значения

глобальных и локальных сложностей во всех направлениях превышают заданные пороговые значения. Если ни одно из условий не выполняется, решение о разбиении не может быть принято и возникает случай 3. Эмпирически подобранное пороговое значение для локальной сложности изображения установлено одно для всех направлений и значений параметра квантования:

$$T_{loc} = 5120 \cdot \left(\frac{3}{4}\right)^d, \tag{21}$$

где d = 0, 1, 2, 3, 4 – текущая глубина разбиений, соответствующая размерам кодируемого блока N = 64, 32, 16, 8, 4. Пороговое значение для глобальной сложности изображения зависит не только от размера кодируемого блока, но и от параметра квантования Qp:

$$T_{\text{glb}} = C(Qp) \cdot \left(\frac{3}{4}\right)^d, \tag{22}$$

где $C(22)$ = 448, $C(27)$ = 704, $C(32)$ = 832, $C(37)$ = 1216, а при других значениях параметра квантования Qp значения $C(Qp)$ определяются путем интерполяции.

В [13] в качестве меры однородности изображения предлагается использовать энтропию квантованных значений интенсивности изображения в кодируемом блоке. Шаг квантования равен 8, так что квантованные значения лежат в диапазоне от 0 до 31. Энтропия по Шеннону рассчитывается по формуле

$$H = -\sum_{i=0}^{31} p_i \cdot \log_2 p_i, \tag{23}$$

где $p_i = \frac{n_i}{N^2}$ – относительная частота значения i среди квантованных значений интенсивности; n_i – количество квантованных значений равных i, N – размер кодируемого блока.

В работе предлагается три условия для принятия решения о делении кодируемого блока на подблоки:

– если энтропия квантованных значений интенсивности в блоке $H < 1{,}2$, то блок не разбивается;
– если $H > 3{,}5$, то кодируемый блок разбивается на подблоки;
– если $H_{avg} - 0{,}15 < H < H_{avg} + 0{,}15$, где H_{avg} – среднее по подблокам всех возможных размеров значение энтропии, то блок не разбивается.

В том случае, когда ни одно из трех условий не выполняется, блок разбивается на подблоки.

Результаты, опубликованные в работах [11–16] и характеризующие эффективность использования перечисленных выше критериев выбора размера блока пространственного предсказания, приведены в таблице 2.

Таблица 2 – Эффективность использования критериев выбора размера блока пространственного предсказания

Название критерия	BD-rate, %	ΔT, %
Kim [8]	0,8	23,8
Cho [9]	2,0	55,8
Shen [10]	1,7	21,1
Yongfei Zhang [11]	4,8	56,7
Min [12]	0,8	52,3
Mengmeng Zhang [13]	3,7	62,0

В ячейках первого столбца таблицы 2 приведены условные названия критериев с указанием ссылки на работу, откуда взяты результаты по их использованию (в качестве условного названия критерия использована фамилия первого автора публикации). Во втором столбце – значения метрики BD-Rate. Величина ΔT, значения которой приведены в третьем столбце таблицы 2, представляет разницу в процентах во времени кодирования между предлагаемым критерием и оценкой по критерию *RDC*. Результаты, представленные авторами, получены с использованием справочной реализации кодера HM.

Наилучшим из рассматриваемых критериев можно считать тот, который обеспечивает минимальное значение величины BD-rate при максимальном значении ΔT. Такой критерий обеспечивает максимальную (среди сравниваемых) степень сжатия при минимальных вычислительных затратах на кодирование. По величине BD-rate наилучшими являются критерии с условными названиями Kim (далее Ким) и Min (далее Мин). Первый из них обеспечивает сокращение объема вычислений всего лишь на 23,8%. Использование критерия Мин обеспечивает сокращение объема вычислений на 52,3%, что делает его наилучшим среди сравниваемых критериев.

3.3 Модификация критерия выбора размера блока пространственного предсказания

Рассмотрим более подробно процедуру выбора размера блока пространственного предсказания, предлагаемую в [12]. После расчета глобальных и локальных сложностей для кодируемого блока решение о разбиении или не разбиении его на подблоки принимается поэтапно. Прежде всего, сложность изображения, рассчитанная для каждого из четырех направлений, сравнивается с пороговым значением. Если в каком-либо из направлений глобальные и локальные сложности всего блока и его подблоков оказываются меньше пороговых значений, то принимается решение о том, что данный блок не требует разбиений. В этом случае размер блока пространственного предсказания оказывается определен уже на первом этапе. На втором этапе минимальные по всем направлениям значения локальных и глобальных сложностей сравниваются с пороговым значением. Если минимальное значение глобальной сложности оказывается больше порога или минимальное значение локальной сложности оказывается больше порога, то принимается решение о том, что блок необходимо разбить на

подблоки. На третьем этапе обрабатываются только те блоки, для которых не удалось принять решение на первых двух этапах. Для этих блоков решение о необходимости разбиения принимается на основании перебора всех возможных вариантов разбиения по минимуму величины *RDC*.

Очевидным вариантом модификации алгоритма Мин [12], обеспечивающим дальнейшее сокращение объема вычислений при выборе размера блока предсказания, является использование какого-либо из альтернативных критериев для блоков, дошедших до третьего этапа рассматриваемого алгоритма. Такая комбинация позволит сократить объем вычислений, так как позволит избежать расчета значений *RDC* для всех возможных вариантов разбиений блоков, обрабатываемых на третьем этапе. С другой стороны, замена полного перебора на альтернативный вариант неизбежно приведет к росту BD-rate, то есть некоторой потере степени сжатия.

В данной работе в качестве такого альтернативного критерия для третьего этапа алгоритма Мин был выбран критерий Ким [8], обеспечивающий минимальную потерю по степени сжатия. Результаты использования комбинированного (Мин и Ким) критерия приведены в таблице 3 [14]. Эксперименты по кодированию проводились на наборе тестовых видеопоследовательностей комитета JCT-VC [26].

В первом столбце таблицы приведены названия тестовых видеопоследовательностей. Во втором столбце указано разрешение видеоизображений. Третий столбец содержит значения величины BD-rate, полученные при сравнении комбинированного алгоритма и оригинального алгоритма Мин. Относительное изменение времени кодирования, вызванное использованием комбинированного алгоритма, приведено в четвертом столбце. В последней строке таблицы представлены средние арифметические значения характеристик BD-rate и ΔT.

Таблица 3 – Эффективность использования комбинированного критерия

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	0,17	6,47
PeopleOnStreet		0,27	6,79
SteamLocomotiveTrain		0,01	2,16
Kimono	1920×1080	0,46	10,94
ParkScene		0,20	3,24
Cactus		0,34	2,14
BQTerrace		–0,02	3,95
BasketballDrive		0,07	7,45
BQMall	832×480	0,02	3,70
PartyScene		0,04	–0,23
BasketballDrill		–0,21	1,09
BQSquare	416×240	0,01	–0,92
BlowingBubbles		–0,02	–1,56
BasketballPass		0,00	4,22
Vidyo1	1280×720	1,10	15,29
Vidyo3		1,20	13,98
Vidyo4		0,84	17,00
BasketballDrillText	832×480	–0,02	2,85
ChinaSpeed	1024×768	0,27	7,19
SlideEditing	1280×720	0,46	17,47
SlideShow		0,88	22,14
В среднем		*0,29*	*6,92*

На основании представленных данных можно сказать, что в среднем введение комбинированного алгоритма сокращает вычислительные затраты

(время кодирования) на 6.92%. При этом возникают потери в степени сжатия видеоданных, которые составляют в данном случае 0.29% в среднем.

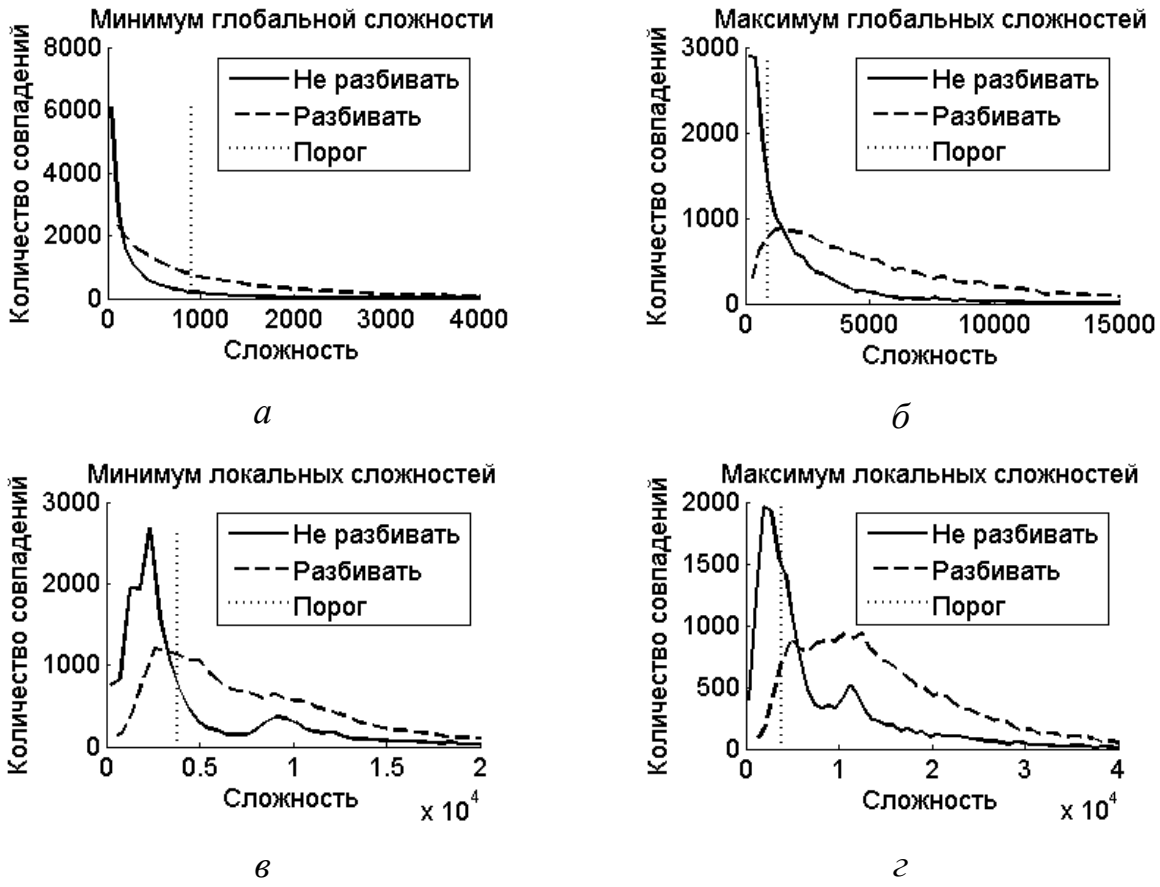
3.4 Регулирование соотношения степени сжатия и вычислительной сложности алгоритмов кодирования

Предлагаемый в данной работе метод определения необходимости разбиения или прекращения разбиения блока кодирования на подблоки, основывается на комбинации двух различных критериев. Оба эти критерия позволяют принять решение на основании того, превышает определенная мера некоторое пороговое значение или нет. Пороговые значения выбраны авторами обоих критериев эмпирическим методом. Рассмотрим влияние предлагаемых пороговых значений на количество правильно принимаемых решений о разбиении и не разбиении.

3.4.1 Исследование влияния пороговых значений критерия Мин на эффективность кодирования

По критерию Мин [12] решение о разбиении или прекращении разбиения блока кодирования принимается на основании величин, называемых глобальными или локальными сложностями. Если максимум этих сложностей не превышает порогового значения, то принимается решение о не разбиении блока. Если минимум этих сложностей превышает пороговое значение, то блок разбивается на подблоки.

На рисунках 6*а–г* представлены полученные в результате вычислений экспериментальные зависимости количества принимаемых решений от посчитанных значений сложности для блоков размером 32×32 пикселя и коэффициента квантования *Qp* равного 37.



а *б*

в *г*

Рисунок 6 – Зависимости количества принимаемых решений от сложности границ (*Qp* – 37, размер блока – 32×32): *а* – от минимума глобальных сложностей, *б* – от максимума глобальных сложностей, *в* – от минимума локальных сложностей, *г* – от максимума локальных сложностей

На графиках по оси абсцисс откладываются числовые значения глобальных (6.*а–б*) или локальных (6.*в–г*) сложностей, по оси ординат – количество принимаемых решений, основанное на статистике, собранной во время кодирования набора видеопоследовательностей. Сплошной и пунктирной линиями изображено количество принимаемых решений о не разбиении и разбиении соответственно, получаемых при выборе варианта разбиения без использования дополнительных критериев, то есть на основании метрики *RDC*. Вертикальной точечной линией изображен порог, предлагаемый автором. Всё, что лежит правее порогового значения на графиках минимальных сложностей (6.*а*, 6.*в*), должно быть разбито. Всё, что лежит

левее порогового значения на графиках максимальных сложностей (6.*а*, 6.*в*), должно оставаться единым блоком.

В данном случае, количество ложных разбиений составляет порядка 11%; количество ложных решений не разбивать блок – 3%; количество верно принятых решений о разбиении и не разбиении составляет 31 и 21% соответственно. В 34% случаев критерий не срабатывает и выполняется полная проверка по метрике *RDC*. Видно, что большое количество ошибок совершается при принятии решения о разбиении блока. При этом количество неверно принимаемых решений о не разбиении мало.

Для устранения этой проблемы в рамках данной работы предлагается разделить пороговые значения для решений «разбивать» и «не разбивать». Было решено держать количество ошибок ложного разбиения на уровне 3%, как и для ошибок ложного не разбиения. В результате аналогичного анализа, проведенного для всех возможных размеров блока кодирования и параметров *Qp* равных 22, 27, 32 и 37, было принято решение принять следующие пороговые значения:

$$T_{loc}^{NS} = T_{loc}, \tag{24.а}$$

$$T_{\text{glb}}^{NS} = T_{\text{glb}}, \tag{24.б}$$

$$T_{loc}^{S} = T_{loc}^{NS} \cdot 2.63, \tag{24.в}$$

$$T_{\text{glb}}^{S} = T_{\text{glb}}^{NS} \cdot 2.63, \tag{24.г}$$

где T_{loc} и T_{glb} – пороговые значения для локальных и глобальных сложностей, предложенные в [12], рассчитываются по формулам (21) и (22) соответственно; T_{loc}^{NS} и T_{glb}^{NS} – пороговые значения для принятия решения о не разбиении на основании значений локальных и глобальных сложностей соответственно; T_{loc}^{S} и T_{glb}^{S} – пороговые значения для принятия решения о разбиении на основании значений локальных и глобальных сложностей соответственно.

Результаты такого подхода отражены в таблицах 4 и 5. В первых столбцах таблиц приведены названия тестовых видеопоследовательностей. Во вторых столбцах указано разрешение видеоизображений. Третий и четвертый столбцы содержат значения величины BD-rate и ΔT соответственно, полученные при сравнении оригинального алгоритма Мин с модификацией, использующей только измененные пороги (таблица 4), и модификацией, использующей как измененные пороги, так и дополнительный критерий Ким.

Таблица 4 – Эффективность использования разделенных порогов без использования дополнительного критерия Ким

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	-1.03	-17.14
PeopleOnStreet		-1.25	-17.28
SteamLocomotiveTrain		-4.65	-16.01
Kimono	1920×1080	-4.96	-22.41
ParkScene		-0.83	-18.83
Cactus		-0.92	-20.29
BQTerrace		-0.65	-16.30
BasketballDrive		-1.49	-23.95
BQMall	832×480	-1.08	-16.41
PartyScene		-0.11	-16.05
BasketballDrill		-0.35	-15.65
BQSquare	416×240	-0.13	-15.61
BlowingBubbles		-0.22	-16.47
BasketballPass		-0.54	-12.49
Vidyo1	1280×720	-3.01	-21.63
Vidyo3		-1.53	-19.17
Vidyo4		-0.9	-18.18
BasketballDrillText	832×480	-0.3	-17.62
ChinaSpeed	1024×768	-0.15	-9.14
SlideEditing	1280×720	-0.08	-11.43
SlideShow		-0.35	-14.98
В среднем		-1.17	-17.00

Из таблицы 4 видно, что при изменении пороговых значений в соответствии с формулами 24.а–г, степень сжатия улучшается более чем на 1%. Вместе с этим происходит увеличение времени кодирования в среднем на 17%. Такое увеличение вычислительных затрат связано с тем, что при разделении одного порогового значения для решений разбиения и не разбиения на два, между ними возникает довольно широкая область, значения сложностей из которой соответствуют полной оценки режима

разбиения по метрике *RDC*. Тот факт, что степень сжатия в данном случае увеличилась более чем на 1%, при том что в [12] авторами заявлено среднее ухудшение показателя на 0,8%, объясняется в разделе 4.2.2.

Таблица 5 – Эффективность использования разделенных порогов с использования дополнительного критерия Ким

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	-0.78	-10.66
PeopleOnStreet		-1.00	-8.19
SteamLocomotiveTrain		-4.59	-8.40
Kimono	1920×1080	-4.17	-5.44
ParkScene		-0.73	-15.09
Cactus		-0.66	-19.94
BQTerrace		-0.56	-6.08
BasketballDrive		-1.42	-3.32
BQMall	832×480	-0.84	-3.36
PartyScene		-0.13	-8.46
BasketballDrill		-0.28	0.83
BQSquare	416×240	-0.21	-5.18
BlowingBubbles		-0.25	-7.41
BasketballPass		-0.12	-7.18
Vidyo1	1280×720	-1.70	-18.46
Vidyo3		-0.27	-14.24
Vidyo4		0.27	-3.01
BasketballDrillText	832×480	-0.32	-2.26
ChinaSpeed	1024×768	0.27	-4.21
SlideEditing	1280×720	0.26	11.39
SlideShow		0.21	14.62
В среднем		-0.81	-5.91

В таблице 5 производится сравнение работы метода Мин с комбинированным методом Мин и Ким и порогами (24). Видно, что используя дополнительный критерий Ким удалось частично компенсировать прирост вычислительной сложности, обусловленный разделением порогов, до 5.91%, сохраняя при этом значительное повышение степени сжатия в среднем на 0.81%.

3.4.2 Исследование влияния пороговых значений критерия Ким на эффективность кодирования

По критерию Ким [8] в зависимости от величины *RDC* принимается решение о необходимости дальнейшего разбиения блока на подблоки. Для этого для каждого размера блока вводятся пороговые значения T_{64}, T_{32}, T_{16}, T_8, рассчитываемые по формулам 12.а–г. Если значение *RDC* текущего уровня разбиения не превышает соответствующий порог, дальнейшее разбиение прекращается.

Оценим количество ошибочных принятий решения о прекращении разбиения для предлагаемых значений порогов. На рисунке 7 представлены полученные в результате вычислений экспериментальные зависимости количества принимаемых решений от посчитанных значений сложности для блоков размером 32×32 пикселя и коэффициента квантования *Qp* равного 37. На графике по оси абсцисс откладываются значения метрики *RDC*, по оси ординат – количество принимаемых решений, основанное на статистике, собранной во время кодирования набора видеопоследовательностей. Сплошной и пунктирной линиями изображено количество принимаемых решений о не разбиении и разбиении соответственно, получаемых при выборе варианта разбиения без использования дополнительных критериев.

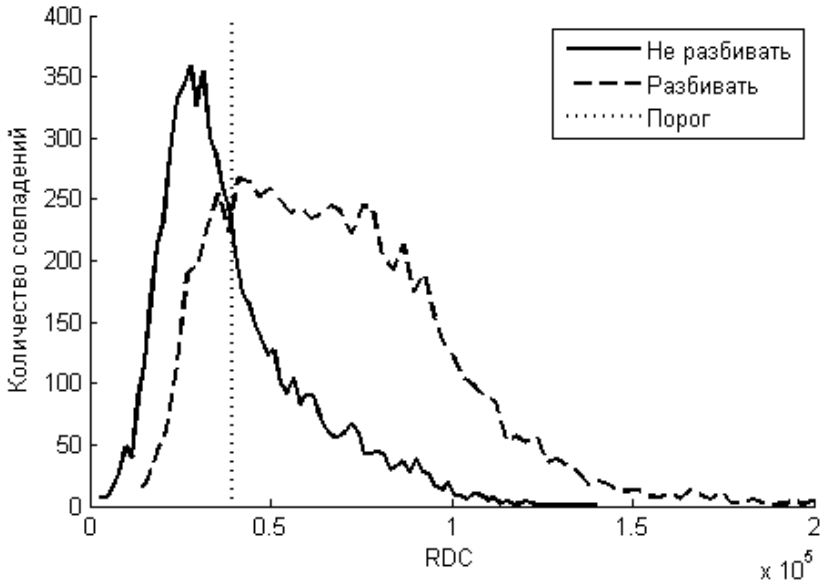


Рисунок 7 – Зависимости количества принимаемых решений от значения метрики *RDC* (*Qp* – 37, размер блока – 32×32)

Вертикальной точечной линией изображен порог, предлагаемый авторами. В соответствии с [8], для всех блоков кодирования, для которых *RDC* лежит на графике левее порогового значения, дальнейшее разбиение прекращается.

Для предлагаемых авторами порогов количество ложных прерываний разбиения составляет 10%, количество правильных прерываний – 30%. В 60% случаев выполняется проверка следующего уровня разбиения.

С целью уменьшения потерь степени сжатия, возникаемых вследствие большого числа ложных прерываний разбиений по критерию Ким, получены новые выражения для расчета пороговых значений [15]:

$$T_{64} = 1265 \cdot e^{0,086 \cdot Qp}, \quad (25.а)$$

$$T_{32} = 179,5 \cdot e^{0,1329 \cdot Qp}, \quad (25.б)$$

$$T_{16} = 26,41 \cdot e^{0,1678 \cdot Qp}, \quad (25.в)$$

$$T_{8} = 1,054 \cdot e^{0,254 \cdot Qp}, \quad (25.г)$$

где *Qp* – коэффициент квантования. Параметры этой зависимости оценены методом наименьших квадратов для эмпирически подобранных значений порогов для *Qp* равных 22, 27, 32 и 37 и для всех возможных размеров блоков кодирования. Для заданных *Qp* пороговые значения подбирались таким образом, чтобы количество ошибочных прерываний разбиений держалось на уровне 2%.

Результаты тестирования эффективности кодирования с использованием новых пороговых значений критерия Ким представлены в таблице 6.

В первом столбце таблицы приведены названия тестовых видеопоследовательностей. Во втором столбце указано разрешение видеоизображений. Третий и четвертый столбцы содержат значения величины BD-rate и ΔT соответственно, полученные при сравнении модифицированного алгоритма Мин + Ким с разделенными порогами по первому критерию и этого же алгоритма с дополнительно измененными

Таблица 6 – Эффективность использования модифицированных порогов дополнительного критерия Ким

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	-0.38	-2.84
PeopleOnStreet		-0.27	-2.62
SteamLocomotiveTrain		-0.54	-6.53
Kimono	1920×1080	-0.11	-0.52
ParkScene		-0.23	-2.53
Cactus		-0.10	-2.86
BQTerrace		-0.54	-5.13
BasketballDrive		-0.13	-0.70
BQMall	832×480	-0.02	-0.17
PartyScene		-0.12	-0.17
BasketballDrill		-0.01	1.88
BQSquare	416×240	-0.01	1.79
BlowingBubbles		-0.07	-0.29
BasketballPass		-0.10	0.17
Vidyo1	1280×720	-0.52	-3.08
Vidyo3		-0.36	2.74
Vidyo4		-2.61	-3.76
BasketballDrillText	832×480	-0.23	-1.15
ChinaSpeed	1024×768	-1.62	-5.01
SlideEditing	1280×720	-1.54	-5.80
SlideShow		-1.15	-6.35
В среднем		-0.51	-2.04

порогами критерия Ким. Из таблицы видно, что понижение порогового значения второго критерия при описанном выше ограничении на число ложных срабатываний может привести к повышению степени компрессии на 0.51% при росте вычислительной сложности в 2.04%.

3.4.3 Обсуждение результатов изменения пороговых значений используемых критериев

Основным недостатком всех методов, позволяющих сократить число перебираемых при кодировании видео вариантов разбиений, является ухудшение степени сжатия. В разделе 4.2.2 показано, объединенный критерий Мин и Ким даёт рост битовой скорости 2.39%. Одной из причин такого эффекта является то, что при заданных значениях порогов часто происходят ошибки первого рода. Для критерия Мин это

ложноположительные решения о необходимости разбиения, для критерия Ким – ложноположительные решения о не разбиении. В подразделах 3.4.1 и 3.4.2 предлагается регулировать число таких ошибок путём изменения пороговых значений.

Для критерия Мин предложено разделить общий порог для решений о разбиении и не разбиении на два отдельных порога. Порог для решения о не разбиении предлагается использовать авторский, а порог для решения о разбиении сдвинуть таким образом, чтобы количество ошибок держалось на уровне, соответствующем количеству ложноположительных срабатываний критерия о не разбиении равному 3%. Для критерия Ким предложено сдвинуть порог таким образом, чтобы количество ошибочных прерываний разбиения держалось на уровне 2%.

Модификация порогов критерия Мин для объединенного критерия позволяет добиться 0.81% повышения степени компрессии. Модифицировав пороги критерия Ким можно улучшить степень сжатия ещё на 0.51%. При этом выигрыш по времени, затрачиваемом на кодирование видеопоследовательности, сокращается на 5.91 и 2.04% соответственно. Таким образом, вычислительную сложность и степень сжатия алгоритмов кодирования можно регулировать изменяя пороговые значения.

3.5 Основные результаты и выводы по разделу

Возможность выбора размера блока пространственного предсказания, введенная в новый стандарт видеокодирования H.265/HEVC, обеспечивает высокое качество предсказания, что приводит к существенному повышению степени сжатия. С другой стороны, полный перебор всех возможных вариантов разбиений кодируемого блока на блоки предсказания для выбора наилучшего варианта приводит к огромным вычислительным затратам. Проведенный в данном разделе обзор известных критериев выбора размера

блока предсказания показал, что наилучшим можно считать критерий Мин [12]. Использование этого критерия позволяет более чем в два раза сократить общее время, затрачиваемое вычислительной системой на кодирование видеоданных. Степень сжатия данных снижается при этом менее чем на 1%. В данном разделе предложены модификации алгоритма Мин, влияющие на вычислительную сложность процесса кодирования видеоизображений и степень сжатия закодированного потока. Основными результатами исследований, описанных в данном разделе, можно считать следующие.

1. На основании обзора статей, посвященных адаптивному разбиению кадров на блоки кодирования в соответствии со стандартом H.265/HEVC, выбран наиболее эффективный метод [12]. Авторы предлагают использовать числовую характеристику, получаемую на основании значений отсчётов изображения. Эта характеристика сравнивается с пороговыми значениями для принятия решения о разбиении или не разбиении блока на подблоки. По измерениям автором, такой подход помогает сократить время кодирования в среднем на 52.3% при понижении степени сжатия на 0.8%. Минусом данного метода является то, что остаются возможными случаи, когда однозначно определить необходимость разбиения только на основании используемой числовой характеристики невозможно. В таком случае решение принимается на основании метрики *RDC*.
2. Предложена модификация метода [12], основанная на критерии, предложенном в [8]. В случае, когда по критерию [12] не удаётся однозначно определить необходимость дальнейшего разбиения, для блока кодирования необходимо вычислять метрику *RDC*. Сравнивая значения этого *RDC* с пороговыми значениями, предложенными в [8], может быть принято решение о том, что дальнейших разбиений блока на подблоки не требуется. Такая модификация позволяет

ускорить процесс кодирования на 6.92% при росте битовой скорости на 0.29% относительно исходного алгоритма [12].

3. С целью повышения эффективности разработанного метода выбора разбиений, предложены варианты модификации используемых в [12] и [8] пороговых значений. Показано, что при разделении порогов критерия [12] для принятия решений о разбиении и решений о не разбиении блоков кодирования можно сэкономить 0.81% роста битовой скорости при повышении вычислительной сложности на 5.92%; при более тщательном подборе пороговых значений критерия [8] можно уменьшить рост битовой скорости ещё на 0.51% при потерях в скорости 2.04%.

Таким образом, предложен новый алгоритм разбиения блока кодирования на подблоки. Кроме того, проведенные исследования показали возможность регулировать соотношение степени сжатия и вычислительной сложности алгоритма разбиения. Такая возможность может оказаться полезной при разработке конкретных реализаций кодирующих систем, удовлетворяющих различным требованиям.

4 Программная реализация быстрых алгоритмов видеосжатия

Данный раздел посвящен разработке и описанию программной реализации кодирующей системы. При разработке за основу была взята версия программной реализации кодера, доступная в исходных кодах [27]. Модификация касалась основной процедуры выбора варианта разбиения изображения на блоки кодирования, что потребовало программной реализации алгоритмов описанных в разделе 3. Оригинальные алгоритмы собраны в библиотеку классов на языке программирования C++, что обеспечивает их универсальность и облегчает практическое использование.

4.1 Описание разработанной библиотеки

Разработанные алгоритмы реализованы на языке программирования C++ в виде класса `TEncSplitter`. Объекты класса `TEncSplitter` могут быть использованы в процессе кодирования видеопоследовательностей на этапе разбиения блоков CTU на дерево блоков кодирования CU для быстрого принятия решения о разбиении.

Открытый (public) интерфейс класса `TEncSplit` включает в себя следующие функции.

- `TEncSplitter()`

  Назначение:

  Конструктор класса. Инициализирует необходимые для работы объекта переменные.

- `Void init(TComYuv *pcYuvSrc, Int iDepth, Int iQP)`

  Входные параметры:

  `pcYuvSrc` – указатель на `TComYuv`. Содержит адрес изображения, соответствующего оцениваемому блоку кодирования.

`iDepth` – целочисленное значение. Принимает значение глубины разбиения оцениваемого блока кодирования.

`iQP` – целочисленное значение. Принимает значение коэффициента квантования *Qp* применяемого для оцениваемого блока.

Назначение:

Инициализатор оценки блока кодирования. Подготавливает данные об анализируемом блоке кодирования, необходимые для принятия решения о разбиении или не разбиении его на подблоки.

- `Void decide()`

  Назначение:

  Данная функция выполняет подсчёт значений сложности, описанных в разделе 3.2 для критерия, предложенного в [12], и принимает решение о необходимости разбиения или не разбиения анализируемого блока кодирования. Полученное решение сохраняется в член класса `m_decision`, имеющий тип-перечисление `SplitterDecision`.

- `SplitterDecision getDecision()`

  Назначение:

  Возвращается член класса `m_decision`, характеризующий принятое решение о необходимости разбиения или не разбиения блока кодирования на подблоки.

- `Void setDepth(Int depth)`

  Входные параметры:

  `Depth` – целочисленное значение. Устанавливаемое значение глубины блока кодирования в квадродереве.

  Назначение:

  Изменяет значение глубины блока кодирования в квадродереве.

- `Void setQP(Int qp)`

Входные параметры:

`qp` – целочисленное значение. Устанавливаемое значение коэффициента квантования.

Назначение:

Изменяет значение коэффициента квантования.

- `Int getDepth()`

  Назначение:

  Возвращает установленное значение глубины блока кодирования в квадродереве.

- `Int getQP()`

  Назначение:

  Возвращает установленное значение коэффициента квантования

- `Double getThresholdRDC()`

  Назначение:

  Возвращает пороговое значение *RDC* для заданных глубины блока кодирования в квадродереве и коэффициента квантования.

Ниже приведены описания скрытых (private) функций класса `TEncSplitter`.

- `Void prepareGlbComplexities()`

  Назначение:

  Функция подсчёта значений глобальных сложностей для принятия решения о разбиении или не разбиении.

- `Void prepareLclComplexities()`

  Назначение:

  Функция подсчёта значений локальных сложностей для принятия решения о разбиении или не разбиении.

- `Int getMeanY(Char filter)()`

  Входные параметры:

`filter` – целочисленное значение. Принимает номер фильтра, который необходимо применить к пикселям для подсчёта среднего арифметического значения яркостей:

«–1» – без фильтра;

«0» – горизонтальный фильтр;

«1» – вертикальный фильтр;

«2» – диагональный фильтр под углом 45 градусов;

«3» – диагональный фильтр под углом 135 градусов.

Значение по умолчанию «–1».

Назначение:

Выполняет подсчёт и возвращает среднее значение яркости пикселей анализируемого блока. При необходимости перед подсчётом выполняется фильтрация значений пикселей по одному из направлений.

– `Int getMeanYSub(Int *vals, Char filter)()`

Входные параметры:

`filter` – целочисленное значение. Принимает номер фильтра, который необходимо применить к пикселям для подсчёта среднего арифметического значения яркостей:

«–1» – без фильтра;

«0» – горизонтальный фильтр;

«1» – вертикальный фильтр;

«2» – диагональный фильтр под углом 45 градусов;

«3» – диагональный фильтр под углом 135 градусов.

Значение по умолчанию «–1».

Выходные параметры:

`vals` – указатель на целочисленный массив длины 4. Заполняется средними арифметическими значениями четырех подблоков анализируемого блока кодирования.

Назначение:

Выполняет подсчёт и возвращает среднее значение яркости пикселей четырех подблоков анализируемого блока кодирования. При необходимости перед подсчётом выполняется фильтрация значений пикселей по одному из направлений.

– `UChar getFilteredValue (Int x, Int y, Char filter)`

Входные параметры:

`x, y` – целочисленные значения. Принимают координаты фильтруемого пикселя.

`filter` – целочисленное значение. Принимает номер фильтра, который необходимо применить к пикселям для подсчёта среднего арифметического значения яркостей:

«–1» – без фильтра;

«0» – горизонтальный фильтр;

«1» – вертикальный фильтр;

«2» – диагональный фильтр под углом 45 градусов;

«3» – диагональный фильтр под углом 135 градусов.

Значение по умолчанию «–1».

Назначение:

Вычисляет и возвращает фильтрованное значение пикселя анализируемого блока кодирования по заданным координатам.

– `static Int getThreasholdGlbForQP(Int qp, UInt depth)`

Входные параметры:

`qp` – целочисленное значение. Принимает коэффициент квантования *Qp*.

`depth` – целочисленное значение. Принимает глубину разбиения в квадродереве. Значение по умолчанию «0».

Назначение:

Вычисляет и возвращает пороговое значение глобальной сложности, соответствующее заданным коэффициенту квантования и глубине разбиения.

– `static Int getThreasholdLclForQP(UInt depth)`

Входные параметры:

`depth` – целочисленное значение. Принимает глубину разбиения в квадродереве. Значение по умолчанию «0».

Назначение:

Вычисляет и возвращает пороговое значение локальной сложности, соответствующее заданной глубине разбиения.

– `Int getMinGlobalComplexity()`

Назначение:

Возвращает минимальное значение из глобальных сложностей по четырём направлениям.

– `Int getMaxGlobalComplexity()`

Назначение:

Возвращает максимальное значение из глобальных сложностей по четырём направлениям.

– `Int getMinLocalComplexity()`

Назначение:

Возвращает минимальное значение из локальных сложностей по четырём направлениям.

– `Int getMaxLocalComplexity()`

Назначение:

Возвращает максимальное значение из локальных сложностей по четырём направлениям.

Далее описывается вспомогательная структура данных:

– `SplitterDecision`

Тип-перечисление. Предназначен для хранения принятого решения о разбиении или не разбиении блока. Возможные значения:

- UNDETERMINED – неопределенное значение, означает, что необходима проверка по критерию *RDC*;

- SPLIT – принято решение о необходимости разбиения блока кодирования на подблоки без оценки *RDC* текущего уровня разбиения;

- NO_SPLIT – принято решение о необходимости прерывания процесса разбиения блока кодирования на подблоки, оценка *RDC* необходима только для текущего блока.

Программный код интерфейса и реализации описанного выше класса представлена в приложении Б.

Для использования разработанной библиотеки в системе кодирования, соответствующей стандарту H.265/HEVC необходимо подключить библиотеку, указав в тексте программы соответствующую директиву `#include "TEncSplitter.h"` и создать объект класса `TEncSplitter`. Далее, в процессе разбиения кадра на блоки кодирования для каждого блока необходимо выполнить инициализацию объекта `TEncSplitter` с помощью функции `init()`, передав ей соответствующие параметры, после чего вызывается функция `decide()`. По завершении её выполнения решение, принятое анализатором, можно получить, вызвав метод `getDecision()`. Если на данном этапе объект `TEncSplitter` не смог принять решения, то есть `getDecision()` возвращает `UNDETERMINED`, то для получения порогового значения метрики *RDC* необходимо вызвать функцию `getThresholdRDC()`.

4.2 Тестирование разработанной библиотеки

Тестирование реализованной библиотеки проводилось на основе справочной реализации кодера HEVC Test Model версии 15.0 (HM15) [22]. Оценка эффективности работы библиотеки производилась путём сравнения исходной реализации HM15 и модифицированной, с использованием библиотеки `TEncSplitter`, и получения оценок BD-rate и ΔT, описанных в разделе 3.1. Все тестирования проводились для настроек кодера, использующих только внутрикадровое кодирование на видеопоследовательностях, длиной в одну секунду.

Замеры результатов работы производились на тестирующей системе, которая имеет следующую конфигурацию:

– процессор Intel Core i5-3330 3.00 ГГц;

– 8192 МБ ОЗУ.

4.2.1 Отладка кодирующей системы

За основу разработанной кодирующей системы была взята версия программной реализации кодера HM15, предоставленная разработчиками стандарта, доступная в исходных кодах [18]. Перед внедрением разработанных алгоритмов в стандартный процесс кодирования, была выполнена сборка и отладка кодера HM15. Проведен анализ её работы и собрана статистика по эффективности кодирования.

На основании анализа процесса кодирования системы HM15 сделаны выводы о том, что при разбиении изображения на блоки кодирования решение принимается на основании перебора всех возможных вариантов. Для каждого блока выполняется оценка метрики *RDC* по формуле (3). Далее рекурсивно выполняется оценка этой метрики для четырёх его подблоков. Если сумма значений *RDC* подблоков меньше метрики *RDC* самого блока, то

принимается решение о разбиении блока кодирования. В противном случае блок не разбивается. Таким образом, система является пригодной для оценки эффективности разработанных алгоритмов.

Для проведения оценки эффективности разработанных алгоритмов производится сравнение производительности кодирующей системы с измеренными характеристиками оригинального алгоритма выбора варианта разбиения кадра на блоки кодирования. С целью получения таких значений выполнено кодирование тестовых видеопоследовательностей на отлаженном кодере HM15 без каких либо модификаций. Проведены замеры затрачиваемого на их кодирование времени и битовой скорости выходного потока для значений коэффициента квантования *Qp* равных 22, 27, 32 и 37.

Таким образом, получен базовый вариант кодирующей системы, к которому можно применять разработанные алгоритмы быстрого поиска оптимального варианта разбиения кадра на блоки кодирования, и собрана статистика по скорости кодирования и степени сжатия базового алгоритма разбиения.

4.2.2 Оценка эффективности кодирования алгоритма Мин

Для оценки эффективности кодирования в соответствии с алгоритмом Мин [12] проведены сравнительные замеры. В таблице 7 приведены результаты сравнения производительности системы кодирования с использованием критерия Мин для быстрого принятия решения о необходимости разбиения или прерывания разбиения с оригинальным кодером HM15, без каких либо модификаций.

В первом столбце таблицы приведены названия тестовых видеопоследовательностей. Во втором столбце указано разрешение видеоизображений.

Таблица 7 – Эффективность использования критерия Мин

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	1.73	47.07
PeopleOnStreet		1.59	44.87
SteamLocomotiveTrain		9.21	41.67
Kimono	1920×1080	6.90	64.55
ParkScene		1.26	50.19
Cactus		1.81	53.54
BQTerrace		1.03	46.55
BasketballDrive		2.10	62.55
BQMall	832×480	1.01	41.08
PartyScene		0.25	32.85
BasketballDrill		2.06	50.22
BQSquare	416×240	0.30	40.91
BlowingBubbles		1.03	37.89
BasketballPass		1.29	44.32
Vidyo1	1280×720	2.02	59.19
Vidyo3		5.16	54.61
Vidyo4		1.69	60.82
BasketballDrillText	832×480	1.11	48.44
ChinaSpeed	1024×768	1.48	53.72
SlideEditing	1280×720	0.29	38.57
SlideShow		0.75	51.44
В среднем		2.10	48.81

Третий и четвертый столбцы содержат значения величины BD-rate и ΔT соответственно, полученные в результате замеров эффективности работы алгоритма Мин. Из таблицы видно, что применение выбранного алгоритма даёт 48.81% снижения вычислительных затрат при ухудшении степени сжатия на 2.10%.

Результаты замеров работы аналогичного алгоритма, представленные в [12] показывают, что авторам удалось добиться снижения вычислительных затрат на 52.3% при ухудшении степени сжатия на 0.8%. Такое расхождение в результатах можно объяснить различной реализацией предлагаемого алгоритма. Например, авторами не поясняется, каким образом подсчитываются значения локальных сложностей. Для их подсчёта необходимо получить фильтрованные значения пикселей, относящихся к блоку кодирования, для чего используются значения соседних пикселей. Для пикселей, лежащих на краях блока кодирования, пиксели из соседних блоков в общем случае не доступны, соответственно получить фильтрованные значения не представляется возможным. В представленной реализации

принято решение опускать пиксели, лежащие на границе рассматриваемого блока кодирования, при подсчёте значений локальных сложностей.

4.2.3 Оценка эффективности модифицированного алгоритма

Для оценки эффективности предложенного алгоритма быстрого принятия решения о необходимости разбиения блока кодирования на подблоки произведен ряд замеров, результаты которых представлены в таблицах 8–10. В первых столбцах таблиц приведены названия тестовых видеопоследовательностей. Во вторых столбцах указано разрешение видеоизображений. Третий и четвертый столбцы содержат значения величины BD-rate и ΔT соответственно, полученные в результате замеров эффективностей работы оцениваемых алгоритмов.

В таблице 8 приводятся результаты оценки скорости кодирования и снижения степени сжатия при использовании комбинированного алгоритма принятия решения о необходимости разбиения блока кодирования на подблоки, описанного в разделе 3.3. Из приведенных в таблице данных видно, что такой подход позволяет снизить вычислительную сложность процесса кодирования видеопоследовательностей в среднем на 51.92% при ухудшении степени сжатия на 2.39%. Сравнивая полученные значения BD-rate и ΔT с результатами раздела 4.2.1 можно сказать, что комбинация критериев Мин и Ким позволяет добиться большего прироста скорости кодирования, при незначительном росте битовой скорости закодированного потока.

Из таблицы 9 видно, что при разделении пороговых значений для принятия решений о разбиении и не разбиении блока кодирования критерия Мин происходит снижение вычислительных затрат в среднем на 45.78% при снижении степени сжатия на 1.54%. Сопоставляя эти значения с данными из таблицы 8, можно заметить, что при таком подходе происходит значительное

Таблица 8 – Эффективность использования комбинированного алгоритма Мин и Ким

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	Δ*T*, %
Traffic	2560×1600	1.91	50.30
PeopleOnStreet		1.85	48.49
SteamLocomotiveTrain		9.23	42.82
Kimono	1920×1080	7.39	67.52
ParkScene		1.47	51.64
Cactus		2.16	54.34
BQTerrace		1.01	48.65
BasketballDrive		2.16	65.10
BQMall	832×480	1.03	43.19
PartyScene		0.29	32.69
BasketballDrill		1.85	50.62
BQSquare	416×240	0.31	40.37
BlowingBubbles		1.01	36.90
BasketballPass		1.28	46.54
Vidyo1	1280×720	3.15	65.23
Vidyo3		6.39	60.76
Vidyo4		2.55	67.26
BasketballDrillText	832×480	1.09	49.83
ChinaSpeed	1024×768	1.75	57.04
SlideEditing	1280×720	0.75	49.32
SlideShow		1.64	62.19
В среднем		2.39	51.94

Таблица 9 – Эффективность использования комбинированного алгоритма Мин и Ким с модифицированными пороговыми значениями критерия Мин

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	Δ*T*, %
Traffic	2560×1600	1.03	42.38
PeopleOnStreet		0.76	40.08
SteamLocomotiveTrain		2.37	52.94
Kimono	1920×1080	1.85	64.14
ParkScene		0.64	42.87
Cactus		0.96	46.56
BQTerrace		0.46	43.40
BasketballDrive		1.17	59.19
BQMall	832×480	1.12	35.01
PartyScene		0.17	24.13
BasketballDrill		1.64	40.90
BQSquare	416×240	0.28	30.20
BlowingBubbles		0.52	28.94
BasketballPass		0.53	38.70
Vidyo1	1280×720	1.30	40.03
Vidyo3		1.82	51.42
Vidyo4		0.86	39.16
BasketballDrillText	832×480	4.61	76.42
ChinaSpeed	1024×768	3.60	37.02
SlideEditing	1280×720	3.57	65.21
SlideShow		3.10	62.66
В среднем		1.54	45.78

Таблица 10 – Эффективность использования комбинированного алгоритма Мин и Ким с модифицированными пороговыми значениями обоих критериев

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	0.66	39.67
PeopleOnStreet		0.49	37.57
SteamLocomotiveTrain		1.25	47.27
Kimono	1920×1080	1.33	58.50
ParkScene		0.53	42.35
Cactus		0.74	44.14
BQTerrace		0.36	40.67
BasketballDrive		0.64	54.58
BQMall	832×480	1.00	34.32
PartyScene		0.15	23.96
BasketballDrill		1.55	40.73
BQSquare	416×240	0.27	32.13
BlowingBubbles		0.51	30.78
BasketballPass		0.46	38.41
Vidyo1	1280×720	1.22	40.20
Vidyo3		1.32	48.51
Vidyo4		0.50	42.04
BasketballDrillText	832×480	2.10	73.12
ChinaSpeed	1024×768	3.50	35.89
SlideEditing	1280×720	2.02	60.77
SlideShow		1.61	57.56
В среднем		1.06	43.96

сокращение роста битовой скорости. Одновременно с этим, значительно уменьшается рост скорости кодирования. Сравнивая эти результаты со статистическими данными, полученными в разделе 4.2.1 можно заметить, что при разделении порогов критерия Мин и комбинировании его с критерием Ким можно значительно снизить степень сжатия видеопоследовательности при не больших потерях скорости кодирования.

Таблица 10 отражает результаты модификации порогов обоих критериев. Снижение вычислительных затрат, по сравнению с не модифицированной реализацией кодера HM15 составляет 43.96%. Целью данной модификации является максимальное сокращение роста битовой скорости, возникающего в результате отсечения проверок некоторого числа вариантов разбиения, среди которых бывают оптимальные варианты. В связи с этим, наибольший интерес представляет значение BD-rate данного варианта алгоритма. Оно составляет 1.06%, что является лучшим результатом среди

всех проведенных замеров. Таким образом, удалось добиться почти двукратного сокращения роста битовой скорости, по сравнению со случаем применения оригинального критерия Мин при небольшой потере роста скорости кодирования.

4.3 Основные результаты и выводы по разделу

Выше описана программная реализации кодирующей системы, основанной на реализации кодера, предоставленная разработчиками стандарта и использующей предложенные в разделе 3 алгоритмы быстрого разбиения кадра на блоки кодирования. Приведены результаты измерений производительности разработанной системы. Основные результаты раздела следующие.

1. Разработана универсальная библиотека классов, реализующая предлагаемые в разделе 3 алгоритмы быстрого принятия решения о необходимости разбиения блоков кодирования на подблоки.
2. Выполнена отладка справочной реализации кодирующей системы HM15 и проведена его модификация. В результате модификации получена система сжатия, использующая разработанную библиотеку классов на этапе поиска оптимального разбиения кадра на блоки кодирования.
3. Проведены замеры с целью оценки эффективности реализованной системы кодирования, с использованием различных модификаций предложенного алгоритма. Оценка эффективности комбинированного алгоритма Мин и Ким, описанного в разделе 3.3, показала снижение времени кодирования в среднем на 51.94% при ухудшении степени сжатия на 2.39%. Показано, что в результате применения модифицированных в соответствии с разделом 3.4 пороговых значений можно повысить степень сжатия

комбинированного алгоритма Мин и Ким. В случае разделения пороговых значений критерия Мин можно добиться снижения степени сжатия на 1.54% при сокращении времени кодирования на 45.78%. В случае модификации порогов обоих критериев снижение степени сжатия составляет 1.06% при ускорении на 43.96%.

Результаты исследования эффективности разработанной системы показывают, что применяя предложенный алгоритм быстрого определения необходимости разбиения блока кодирования на подблоки, можно добиться двукратного сокращения вычислительных затрат на кодирование видео. С другой стороны, при таком подходе происходит ухудшение степени сжатия. Показана возможность регулирования соотношения снижения степени сжатия и сокращения затрачиваемого на кодирование времени. В результате, получена быстрая система сжатия видеоданных, соответствующая стандарту H.265/HEVC, позволяющая настраивать процесс сжатия в зависимости от конкретных требований.

5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

В данном разделе проводится анализ перспективности проведения проектных исследований, планирование комплекса работ, определение трудоёмкости и длительности исполнения, расчет сметы на выполнение проекта и описывается оценка экономической эффективности проекта.

5.1 Организация и планирование работ

При организации процесса реализации проекта необходимо рационально запланировать занятость каждого из его участников (научный руководитель и исполнитель) и сроки проведения отдельных работ. В таблице 11 представлен перечень работ и продолжительность их выполнения во время выполнения данной диссертации.

Таблица 11 – Перечень работ и продолжительность их выполнения

Этапы работы	Исполнители	Загрузка исполнителей
Постановка целей и задач, получение исходных данных	НР	НР – 100%
Формирование и составление требований к проекту	НР, И	НР – 100% И – 30%
Подбор и изучение материалов по тематике, анализ предметной области	НР, И	НР – 30% И – 100%
Разработка календарного плана	НР, И	НР – 100% И – 30%
Обсуждение литературы	НР, И	НР – 70% И – 100%
Разработка на основе разработанных ранее методов новых методов и алгоритмов	НР, И	НР – 100% И – 100%
Реализация полученных алгоритмов на языке программирования C++	И	И – 100%
Анализ полученных результатов	НР, И	НР – 80% И – 100%
Оформление расчетно-пояснительной записки	И	И – 100%
Подведение итогов	НР, И	НР – 60% И – 100%

5.1.1 Продолжительность этапов работ

Расчет продолжительности этапов работ при выполнении магистерской диссертации является важным этапом, так как мы можем определить трудоемкость проводимых работ, а трудовые затраты составляют основную часть стоимости научно-исследовательской работы (НИР).

Под трудоемкостью работ понимают максимально допустимые затраты труда в человеко-днях на выполнение НИР с учетом организационно технических мероприятий, обеспечивающих наиболее рациональное использование выделенных ресурсов.

Существуют различные методы расчета продолжительности этапов работы. В рамках данной НИР используется экспертный способ. Он предполагает генерацию необходимых количественных оценок специалистами конкретной предметной области, опирающимися на их профессиональный опыт и эрудицию.

Для определения вероятных (ожидаемых) значений продолжительности работ $t_{ож}$ применяется следующая формула.

$$t_{ож} = \frac{3 \cdot t_{min} + 2 \cdot t_{max}}{5}, \quad (26)$$

где t_{min} – минимальная продолжительность работы, дн.; t_{max}– максимальная продолжительность работы, дн.

Для выполнения перечисленных в таблице 2 работ требуются специалисты:

- математик-программист – в его роли действует исполнитель НИР;
- научный руководитель.

Для построения линейного графика необходимо рассчитать длительность этапов в рабочих днях, а затем перевести ее в календарные дни.

Расчет продолжительности выполнения каждого этапа в рабочих днях ($T_{РД}$) ведется по формуле:

$$T_{РД} = \frac{t_{ож}}{K_{ВН}} \cdot K_{Д}, \quad (27)$$

где $t_{ож}$ – продолжительность работы, дн.; $K_{вн}$ – коэффициент выполнения работ, учитывающий влияние внешних факторов на соблюдение предварительно определенных длительностей; $K_{Д}$ – коэффициент, учитывающий дополнительное время на компенсацию непредвиденных задержек и согласование работ.

Расчет продолжительности этапа в календарных днях ведется по формуле:

$$T_{КД} = T_{РД} \cdot T_{К}, \quad (28)$$

где $T_{КД}$ – продолжительность выполнения этапа в календарных днях; $T_{К}$ – коэффициент календарности, позволяющий перейти от длительности работ в рабочих днях к их аналогам в календарных днях, и рассчитываемый по формуле:

$$T_{К} = \frac{T_{КАЛ}}{T_{КАЛ} - T_{ВД} - T_{ПД}}, \quad (29)$$

где $T_{КАЛ}$ – календарные дни ($T_{КАЛ}$ = 365);$T_{ВД}$ – выходные дни ($T_{ВД}$ = 52); $T_{ПД}$ – праздничные дни ($T_{ПД}$ = 10). По формуле (4) рассчитаем:

$$T_{К} = \frac{365}{365 - 52 - 10} = 1{,}2 \quad (30)$$

В таблице 12 приведен расчет определения продолжительности этапов работ и их трудоемкости по исполнителям, занятым на каждом этапе. По показанию полученных величины трудоемкости этапов по исполнителям $T_{КД}$ построен линейный график осуществления проекта (таблица 13).

Таблица 12 – Расчет трудозатрат на выполнения проекта

Этап	Исполнители	Продолжительность работ, дни			Трудоемкостьработ по исполнителям чел.- дн.			
					$T_{РД}$		$T_{КД}$	
		t_{min}	t_{max}	$t_{ож}$	НР	И	НР	И
Постановка целей и задач, получение исходных данных	НР	1	2	1,4	1,54	–	1,85	–
Формирование и составление требований к проекту	НР, И	1	3	1,8	1,98	0,59	2,4	0,71
Подбор и изучение материалов по тематике, анализ предметной области	НР, И	5	10	7	2,31	7,7	2,77	9,24
Разработка календарного плана	НР, И	2	4	2,8	3,08	0,92	3,7	1,11
Изучение и обсуждение литературы	НР, И	7	14	9,8	7,55	10,78	9,06	12,94
Разработка на основе разработанных ранее методов новых методов и алгоритмов	НР, И	10	16	12,4	13,6	13,6	16,3	16,3
Реализация полученных алгоритмов на языке программирования С++	И	8	14	10,4	–	11,4	–	15,9
Анализ полученных результатов	НР, И	15	20	17	14,96	18,7	17,95	22,44
Оформление расчетно-пояснительной записки	И	6	9	7,2	–	7,92	–	9,5
Подведение итогов	НР, И	5	8	6,2	4,09	6,82	4,91	8,19
Итого:				76	49,11	78,43	58,94	96,33

Таблица 13 – Линейный график работ

Этап	**НР**	**И**	Март			Апрель			Май			Июнь	
			10	20	30	40	50	60	70	80	90	100	110
1	1,85	–											
2	2,4	0,71											
3	2,77	9,24											
4	3,7	1,11											
5	9,06	12,94											
6	16,3	16,3											
7	–	15,9											
8	17,95	22,44											
9	–	9,5											
10	4,91	8,19											

НР – ; И – .

5.1.2 Расчет накопления готовности проекта

Величина накопления готовности работы показывает, на сколько процентов по окончании текущего (*i*-го) этапа выполнен общий объем работ по проекту в целом. Степень готовности определяется по формуле:

$$СГ_i = \frac{TP_i^H}{TP_{общ.}} = \frac{\sum_{k=1}^{i} TP_k}{TP_{общ.}} = \frac{\sum_{k=1}^{i}\sum_{j=1}^{m} TP_{km}}{\sum_{k=1}^{I}\sum_{j=1}^{m} TP_{km}}, \quad (31)$$

где $TP_{общ.}$ – общая трудоемкость проекта; TP_i (TP_k) − трудоемкость *i*-го (*k*-го) этапа проекта, $i = \overline{1,I}$; TP_i^H − накопленная трудоемкость *i*-го этапа проекта по его завершении; TP_{ij} (TP_{kj}) − трудоемкость работ, выполняемых *j*-м

участником на *i*-м этапе, здесь $j = \overline{1,m}$ − индекс исполнителя, в нашем примере *m* = 2.

В таблице 14 –представлены расчеты нарастания технической готовности работы и удельного веса каждого этапа.

Таблица 14 – Нарастание технической готовности работы

Этап	$ТР_i$, %	$СГ_i$, %
Постановка задачи	1.19	1.19
Разработка и утверждение технического задания (ТЗ)	2.00	3.19
Подбор и изучение материалов по тематике	7.73	10.93
Разработка календарного плана	3.10	14.03
Обсуждение литературы	14.17	28.20
Выбор структурной схемы устройства	21.00	49.19
Выбор принципиальной схемы устройства	10.24	59.43
Расчет принципиальной схемы устройства	26.01	85.44
Оформление расчетно-пояснительной записки	6.12	91.56
Подведение итогов	8.44	100.00

5.2 Расчет сметы затрат на выполнение проекта

В состав затрат на создание проекта включается величина всех расходов, необходимых для реализации комплекса работ, составляющих содержание данной разработки. Расчет сметной стоимости ее выполнения производится по следующим статьям затрат:

- материалы и покупные изделия;
- заработная плата;
- социальный налог;

- расходы на электроэнергию (без освещения);
- амортизационные отчисления;
- оплата услуг связи;
- прочие (накладные расходы) расходы.

5.2.1 Расчет затрат на материалы

Величина материальных затрат определяется исходя из стоимости всех материалов, используемых при выполнении магистерской диссертации, в том числе расходы на их приобретение. В таблице 15 представлен расчет затрат на используемые материалы.

Таблица 15 – Расчет затрат на используемые материалы

Наименование материалов	Цена за ед., руб.	Кол-во	Сумма, руб.
Бумага для принтера формата А4	190	1 уп.	150
Картридж для принтера	1550	1 шт.	1550
Итого:			1700

Транспортно-заготовительные расходы (ТЗР) составляют 5 % от отпускной цены материалов, тогда расходы на материалы с учетом ТЗР равны:

$$C_{мат} = 1700 \cdot 1{,}05 = 1785 \text{ руб.}$$

5.2.2 Расчет заработной платы

Смета затрат на оплату труда в большинстве случаев составляет наибольшую часть себестоимости ВКР. Среднедневная тарифная заработная плата ($ЗП_{дн\text{-}т}$) рассчитывается по формуле:

$$ЗП_{дн-т} = \frac{МО}{21{,}83}, \quad (32)$$

учитывающей, что в году 298 рабочих дней и, следовательно, в месяце в среднем 21,83 рабочих дня (при шестидневной рабочей неделе).

Расчеты затрат на полную заработную плату приведены в таблице 16. Затраты времени по каждому исполнителю в рабочих днях с округлением до целого взяты из таблицы 2. Для учета в ее составе премий, дополнительной зарплаты и районной надбавки используется следующий ряд коэффициентов: $К_{ПР}$= 1,1; $К_{доп.ЗП}$ = 1,113; $К_{р}$ = 1,3.

Таким образом, для перехода от тарифной (базовой) суммы заработка исполнителя, связанной с участием в проекте, к соответствующему полному заработку (зарплатной части сметы) необходимо первую умножить на интегральный коэффициент $К_{и}$ = 1,1· 1,113·1,3 = 1,62 (для пятидневной рабочей недели). Для научного руководителя этот коэффициент составит $К_{и}$ = 1,1· 1,1·1,3 = 1,699 (для шестидневной рабочей недели)

Таблица 16 – Расчет затрат на полную заработную плату

Исполнитель	Оклад, руб./мес.	Среднедневная ставка, руб./раб.день	Затраты времени, раб.дни	Коэффициент	Фонд з/платы, руб.
НР	23 264,86	936,97	50	1,699	79595.60
И	30 000	1428,57	79	1,62	182828.39
Итого:					262423.99

5.2.3 Расчет затрат на социальный налог

Затраты на единый социальный налог (ЕСН), включающий в себя отчисления в пенсионный фонд, на социальное и медицинское страхование, составляют 30 % от полной заработной платы по проекту, то есть:

$$С_{соц.} = С_{зп}·0.3.$$

Итак, в нашем случае:

$$C_{соц.} = 262423.99* 0.3 = 78727.20 \text{ руб.}$$

5.2.4 Расчет затрат на электроэнергию

Данный вид расходов включает в себя затраты на электроэнергию, потраченную в ходе выполнения проекта на работу используемого оборудования, рассчитываемые по формуле:

$$C_{эл.об.} = P_{об} \cdot t_{об} \cdot Ц_Э, \quad (33)$$

где $P_{ОБ}$ – мощность, потребляемая оборудованием, кВт; $Ц_Э$ – тариф на 1 кВт·час; $t_{об}$ – время работы оборудования, час.

В Томском политехническом университете $Ц_Э$ = 5,257 руб./квт·час с учетом налога на добавленную стоимость.

Время работы оборудования вычисляется на основе итоговых данных таблицы 2 для инженера ($T_{РД}$) из расчета, что продолжительность рабочего дня равна 8 часов.

$$t_{об} = T_{РД} \cdot K_t, \quad (34)$$

где $K_t \leq 1$– коэффициент использования оборудования по времени, равный отношению времени его работы в процессе выполнения проекта к $T_{РД}$, определяется исполнителем самостоятельно.

Мощность, потребляемая оборудованием, определяется по формуле:

$$P_{ОБ} = P_{ном.} \cdot K_C, \quad (35)$$

где $P_{ном.}$ – номинальная мощность оборудования, кВт; $K_C \leq 1$ –коэффициент загрузки, зависящий от средней степени использования номинальной мощности. Для технологического оборудования малой мощности $K_C = 1$.

Расчет затраты на электроэнергию для технологических целей представлены в таблице 7.

Таблица 7 –Затраты на электроэнергию технологическую

Наименование оборудования	Время работы оборудования $t_{ОБ}$,час	Потребляемая мощность $P_{ОБ}$, кВт	Затраты $Э_{ОБ}$, руб.
Персональный компьютер	632·0,6	0,3	598.04
Струйный принтер	8	0,1	4.21
Итого:			602.25

5.2.5 Расчет амортизационных расходов

В данном разделе рассчитывается амортизация используемого оборудования за время выполнения проекта по формуле:

$$C_{АМ} = \frac{H_А \cdot Ц_{об} \cdot t_{рф} \cdot n}{F_Д}, \quad (36)$$

где $H_А$ – годовая норма амортизации единицы оборудования; $Ц_{ОБ}$ – балансовая стоимость единицы оборудования с учетом ТЗР. При невозможности получить соответствующие данные из бухгалтерии она может быть заменена действующей ценой, содержащейся в ценниках, прейскурантах и т.п.; $F_Д$ – действительный годовой фонд времени работы соответствующего оборудования, берется из специальных справочников или фактического режима его использования в текущем календарном году. При этом второй вариант позволяет получить более объективную оценку $C_{АМ}$; n – число задействованных однотипных единиц оборудования.

Рассчитаем амортизацию используемого компьютера по формуле (10):

$$C_{АМ} = \frac{0.4 \cdot 40000 \cdot 632 \cdot 1}{2384} = 4241.61 руб.$$

Рассчитаем амортизацию используемого принтера по формуле (10):

$$C_{АМ} = \frac{0,5 \cdot 10000 \cdot 8 \cdot 1}{500} = 80 руб.$$

5.2.6 Расчет прочих расходов

В статье «Прочие расходы» отражены расходы на выполнение проекта, которые не учтены в предыдущих статьях, к ним относятся содержание оргтехники, услуги связи, представительные расходы и другие. Их следует принять равными 10% от суммы всех предыдущих расходов:

$$C_{проч.} = (C_{мат} + C_{зп} + C_{соц} + C_{эл.об.} + C_{ам}) \cdot 0{,}1 \qquad (37)$$

Найдем прочие расходы по формуле (11) учитывая данные полученные выше:

$$C_{проч.} = (1785 + 262423.99 + 78727.20 + 602.25 + 4321.61) \cdot 0.1 = 34786.0.$$

5.2.7 Расчет общей себестоимости разработки

Проведя расчет по всем статьям сметы затрат на разработку, можно определить общую себестоимость проекта «Разработка и исследование методов сжатия сложных сигналов на основе оптимальной и субъоптимальной обработки фазо-частотных характеристик». Смета затрат на разработку представлена в таблице 18.

Таблица 18 – Смета затрат на разработку проекта

Статья затрат	Условное обозначение	Сумма, руб.
Материалы и покупные изделия	$C_{мат}$	1785
Основная заработная плата	$C_{зп}$	262423.99
Отчисления в социальные фонды	$C_{соц}$	78727.20
Расходы на электроэнергию	$C_{эл.}$	602.25
Амортизационные отчисления	$C_{ам}$	4321.61
Прочие расходы	$C_{проч}$	34786.00
Итого:		382646.05

Таким образом, затраты на разработку составили C = 382646.05 руб.

5.2.8 Расчет прибыли

Прибыль от реализации проекта в зависимости от конкретной ситуации (масштаб и характер получаемого результата, степень его определенности и коммерциализации, специфика целевого сегмента рынка и т.д.) может определяться различными способами. В данной работе исполнитель не располагает данными для применения «сложных» методов, отсюда прибыль следует принять в размере 5 ÷ 20 % от полной себестоимости проекта. Таким образом она составляет 76529.21 руб. (20 %) от расходов на разработку проекта.

5.2.9 Расчет НДС

НДС составляет 18% от суммы затрат на разработку и прибыли. В нашем случае это (382646.05 + 76529.21) · 0.18 = 82651.55 руб.

5.2.10 Цена разработки НИР

Цена равна сумме полной себестоимости, прибыли и НДС, в нашем случае

$$Ц_{НИР(КР)} = 382646.05 + 76529.21 + 82651.55 = 541826.81 \text{ руб.}$$

5.3 Оценка экономической эффективности проекта

Актуальным аспектом качества выполненного проекта является экономическая эффективность его реализации, то есть соотношение обусловленного ей экономического результата (эффекта) и затрат на разработку проекта.

В рамках данной работы оценка экономической эффективности не представляется возможной, так как разработанные и реализованные методы и алгоритмы не являются самостоятельным рыночным продуктом. Потребителем эффекта является заказчик – компания «Элекард», финансовая и коммерческая информация о которой является закрытой для меня.

5.3.1 Оценка научно-технического уровня НИР

Научно-технический уровень характеризует влияние проекта на уровень и динамику обеспечения научно-технического прогресса в области обработки сейсмических сигналов. Для оценки научной ценности, технической значимости и эффективности, планируемых и выполняемых научно-исследовательскую работу, используется метод балльных оценок. Балльная оценка заключается в том, что каждому фактору по принятой шкале присваивается определенное количество баллов. Обобщенную оценку проводят по сумме баллов по всем показателям. На ее основе делается вывод о целесообразности работы.

Сущность метода заключается в том, что на основании оценок признаков работы определяется интегральный показатель (индекс) ее научно-технического уровня по формуле:

$$K_{НТУ} = \sum_{i=1}^{3} R_i \cdot n_i, \qquad (38)$$

где $K_{НТУ}$ – интегральный индекс научно-технического уровня; R_i – весовой коэффициент i-го признака научно-технического эффекта; n_i – количественная оценка i-го признака научно-технического эффекта, в баллах.

Так как все частные признаки научно-технического уровня оцениваются по 10-балльной шкале, а сумма весов Ri равна единице, то величина интегрального показателя также принадлежит интервалу [0, 10].

Таблица 19 – Весовые коэффициенты признаков НТУ

Признаки научно-технического эффекта НИР	Характеристика признака НИР	R_i
Уровень новизны	Математический аппарат не является принципиально новым, но содержит ряд модернизированных функций и методов.	0,3
Теоретический уровень	Разработан собственный алгоритм определения оптимального разбиения кадра на блоки кодирования и реализован на языке программирования C++.	0,4
Возможность реализации	Реализованный алгоритм позволит увеличить скорость кодирования видеопоследовательностей в системах кодирования, основанных на стандарте H.265/HEVC.	0,3

В таблице 20 представлены частные оценки уровня n_i и их краткое обоснование.

Таблица 20 – Оценки научно-технического уровня научно-исследовательской работы

Значимость	Фактор НТУ	Уровень фактора	Выбранный балл	Обоснование выбранного балла
0,3	Уровень новизны	Относительно новая	7	Разработанный алгоритм адаптивного разбиения кадра на блоки основан на предложенных ранее быстрых разбиений, но в такой комбинации является новым.
0,4	Теоретический уровень	Разработка метода и алгоритма	7	Разработан и реализован алгоритм адаптивного разбиения кадра на блоки кодирования в рамках H.265/HEVC
0,3	Возможность реализации	В течение первых лет	10	Т.к. алгоритм реализован в виде отдельной библиотеки, он может быть использован в любой кодирующей системе, соответствующей стандарту H.265/HEVC

Отсюда интегральный показатель научно-технического уровня для данного проекта составляет рассчитанный по формуле (38):

$$K_{\text{нту}} = 0{,}3 \cdot 7 + 0{,}4 \cdot 7 + 0{,}3 \cdot 10 = 7{,}9.$$

Таким образом, данная разработка достаточно эффективна, в первую очередь из-за высокой научной значимости и актуальности.

6 Социальная ответственность

Данный раздел посвящен обсуждению производственной безопасности, охраны окружающей среды, защиты в чрезвычайных ситуациях, правовым и организационным вопросам.

6.1 Производственная безопасность

Перечень вредных и опасных факторов, возникающих в ходе разработки методов и алгоритмов по ГОСТ 12.0.003-74 «Опасные и вредные производственные факторы» [28], представлены в таблице 21.

Таблица 21 – Опасные и вредные факторы при разработке алгоритмов на ЭВМ

Источник фактора, наименование видов работ	Факторы (по ГОСТ 12.0.003-74 [1])		Нормативные документы
	Вредные	Опасные	
1. Анализ и разработка математической модели. 2. Описание алгоритмов исследования. 3. Выбор необходимого оборудования для реализации алгоритма. 4. Реализация алгоритма.	1. Повышенный уровень шума; 2. Несоответствующий микроклимат рабочего помещения; 3. Повышенный уровень электромагнитных излучений; 4. Недостаточная освещенность рабочего помещения;	1. Электрический ток.	1. Параметры микроклимата устанавливаются СанПиН 2.2.4-548-96. 2. Параметры излучений дисплеев и шума устанавливаются СанПиН 2.2.2/2.4.1340–03. 3. Рабочее место должно соответствовать требованиям ГОСТ 12.2.032-78 и СанПиН 2.2.2/2.4.1340-03. 4. Освещение устанавливается СП 52.13330.211 и СанПиН 2.2.2/2.4.1340–03.

Рассмотрим эти факторы более подробно.

Производственный шум. Шум – это совокупность звуков, неблагоприятно воздействующих на организм человека и мешающих его работе и отдыху. Основным источником шума при работе с вычислительными машинами является системный блок. Современные процессоры могут включать в себя несколько ядер и дополнительных сопроцессоров. С одной стороны, это повышает производительность системы, с другой – повышается энергопотребление и тепловыделение. Вследствие этого, возникает необходимость в более мощном охлаждении. Для этого современные системные блоки используют всё более мощные системы вентиляции. Соответственно уровень шума, создаваемый системой вентиляции, растет. Кроме процессора в системном блоке используются другие элементы: блок питания, графические платы, жесткие диски – требующие отдельной системы вентиляции, что ещё больше повышает уровень шума.

Шум оказывает отрицательное влияние, как на качество работы человека, так и на его здоровье. Постоянное воздействие сильного шума может не только отрицательно повлиять на слух, но и вызвать другие вредные последствия – звон в ушах, головокружение, головную боль, повышение усталости. Человек, постоянно подвергающийся воздействию шума, быстро переутомляется, отличается повышенной раздражительностью, становится забывчивым, чаще страдает от слабости и головокружения. Уровень шума на рабочих местах, связанных с работой на ПЭВМ, не должен превышать 50 дБА [29].

Требования к параметрам шума устанавливаются СанПин 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки» [29].

Меры, которые необходимо принять, для того чтобы помещение было менее зашумленным, в первую очередь направлены на обеспечение

нормальной вентиляцию системного блока. Для охлаждения необходимо обеспечить со стороны вентиляционных отверстий хотя бы 20-30 см свободного пространства. Не загромождать оборудование посторонними предметами, которые снижают теплоотдачу, периодически прочищать вентиляционные отверстия от пыли.

Несоответствующий микроклимат рабочего помещения. Требования к параметрам микроклимата и воздушной среды определяются согласно СанПиН 2.2.548-96 «Гигиенические требования к микроклимату производственных помещений» [30].

Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (значительные или незначительные тепловыделения). Допустимые параметры микроклимата для категории работ, к которой относится работа математика программиста, приведены в таблице 22.

В настоящее время для обеспечения комфортных условий для работы программиста математика, рекомендуется использовать технические средства, которые включают вентиляцию, кондиционирование воздуха, отопительную систему.

Таблица 22 – Допустимые и оптимальные значения параметров микроклимата

Время года	Зона	Температура воздуха, С	Относительная влажность, %	Скорость движения воздуха, м/с
Холодный период	Оптимальная	18 – 21	60 – 40	< 0.2
Теплый период года (t> 100С)	Оптимальная	20 – 25	60 – 40	< 0.3
	Допустимая	< 28 в 13 часов самого жаркого месяца	< 75	< 0.5

Повышенный уровень электромагнитных излучений. Когда все устройства персонального компьютера включены, в районе рабочего места,

формируется сложное по структуре электромагнитное поле. Реальную угрозу для пользователя компьютера представляют электромагнитные поля, которые создаются за счет работы монитора и системного блока компьютера.

Влияние их на организм человека не обходится без последствий. Исследования показали, что в организме человека под влиянием электромагнитного излучения монитора происходят значительные изменения гормонального состояния, специфические изменения биотоков головного мозга, изменение обмена веществ. Пыль, притягиваемая электростатическим полем монитора, иногда становится причиной дерматитов лица, обострения астматических симптомов, раздражения слизистых оболочек.

Требования и нормы на параметры излучений дисплеев включены в СанПиН 2.2.2/2.4.1340–03 [31].

Для снижения воздействия электромагнитного излучения следует применять мониторы с пониженным уровнем излучения, придерживаться регламентированного режима труда и отдыха, а также проводить регулярную гигиеническую уборку помещения.

Несоответствующая освещенность рабочего помещения. В лабораториях, оснащенных ЭВМ должно быть естественное и искусственное освещение, то есть совмещенное. Естественное освещение обеспечивается через оконные проемы с коэффициентом естественного освещения (КЕО) не ниже 1,2% в зонах с устойчивым снежным покровом и не ниже 1,5% на остальной территории. Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева. Искусственное освещение в помещениях с компьютерами должно осуществляться системой общего равномерного освещения. Освещенность на поверхности стола в зоне размещения документа должна быть 300-500 лк. Допускается установка светильников местного освещения для подсветки документов. Местное освещение не

должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк [32].

Более того, основным гигиеническим требованием является достаточно равномерная освещенность всего поля зрения. То есть уровень освещенности помещения, и яркость экрана монитора должны быть соотносимы: яркий свет в районе периферийного зрения повышает напряженность глаз и приводит к утомляемости.

Электрический ток. Во время использования средств вычислительной техники или других периферийных устройств работник должен осторожно обращаться с электропроводкой и компьютером, а также должен всегда помнить, что, если не придерживаться правил безопасности, то это может угрожать его здоровью.

Чтобы избежать поражения электрическим током, необходимо выполнять следующие правила:

1. Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. При обнаружении неисправности немедленно обесточить электрооборудование. Продолжение работы возможно только после устранения неисправности.

2. Для исключения поражения электрическим током запрещается:

а) часто включать и выключать компьютер без необходимости;

б) прикасаться к экрану и к тыльной стороне блоков компьютера;

в) работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;

г) работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе

д) класть на средства вычислительной техники и периферийном оборудовании посторонние предметы.

3. Запрещается под напряжением очищать от пыли и загрязнения электрооборудование.

4. Ремонт электроаппаратуры производится только специалистами-техниками с соблюдением необходимых технических требований [33].

Во всех случаях поражения человека электрическим током необходимо немедленно вызвать врача. До прибытия врача нужно, не теряя времени, приступить к оказанию первой помощи пострадавшему.

6.2 Охрана окружающей среды

Одной из основных проблем в современном мире является загрязнение литосферы. В первую очередь она загрязняется твердыми отходами, которые накапливаются на свалках, в отвалах и являются опасными источниками загрязнения земной поверхности. Бытовой мусор состоит из бумаги, металла, древесины, стекла, полимеров и др. Основными инструментами при работе над разработкой новых методов и алгоритмов являются компьютерная техника, бумага, люминесцентные лампы и т.д. По истечению срока службы они становятся бытовым мусором. Рассмотрим загрязнения литосферы бытовым мусором, на примере люминесцентных ламп. Их эксплуатация требует осторожности и четкого выполнения инструкции по обращению с данным отходом (код отхода 35330100 13 01 1, класс опасности – 1 [34]). В данной лампе содержится опасное вещество – ртуть в газообразном состоянии. При неправильной утилизации, лампа может разбиться и пары ртути могут попасть в окружающую среду. Вдыхание паров ртути может привести к тяжелому повреждению здоровья.

При перегорании ртутьсодержащей лампы (выходе из строя) её замену осуществляет лицо, ответственное за сбор и хранение ламп (обученное по

электробезопасности и правилам обращения с отходом). Отработанные люминесцентные лампы сдаются только на полигон токсичных отходов для захоронения. Запрещается сваливать отработанные люминесцентные лампы с бытовым мусором [35].

Бытовой мусор помещений организаций: несортированный, образованный в результате деятельности работников предприятия (код отхода 91200400 01 00 4); агрегатное состояние отхода – твердое; основные компоненты: бумага и древесина, металлы, пластмассы и др [34]. Для сбора мусора рабочее место оснащается урной. При заполнении урны, мусор выносится в контейнер бытовых отходов. Предприятие заключает договор с коммунальным хозяйством по вывозу и размещению мусора на организованных свалках.

6.3 Защита в чрезвычайных ситуациях

Чрезвычайной ситуацией (ЧС) называют сложившуюся ситуацию на определенной территории, которая произошла вследствие стихийного бедствия или аварии, которые могут привести к ущербу здоровья людей или окружающей среды, а также к большим материальным потерям.

Наиболее вероятной ЧС в рамках выполнения данной работы является пожар [36].

К мерам по предупреждению данного вида ЧС отнесем следующие пожарно-профилактические мероприятия - соблюдение эксплуатационных норм оборудования, обучение персонала правилам техники безопасности; издание противопожарных инструкций, планов эвакуации.

Профилактические методы борьбы с пожарами в помещении предусматривают:

- организационные методы: надлежащее содержание помещений, инструктаж персонала;

- технические методы: соблюдение противопожарных правил;
- эксплуатационные методы: своевременная профилактика, ремонт оборудования [37].

Основными мерами по повышению устойчивости помещения к данной ЧС являются в первую очередь исключение образования благоприятной для пожара среды (контроль воздухообмена, вентиляция), а также использование трудносгораемых материалов при отделке рабочего помещения.

Необходимыми действиями в результате возникшей ЧС и мерами по ликвидации её последствий являются:

1. Передать сигнал «Тревога» голосом, задействовать систему оповещения людей о пожаре.
2. Сообщить по телефону 01, с сотового 010 адрес объекта, место возникновения пожара, свою фамилию. Сообщить по телефону 03, с сотового 030 адрес объекта, что случилось, информацию о пострадавших, свою фамилию, оказать помощь пострадавшим.
3. Направить людей к эвакуационным выходам согласно знакам направления движения.
4. Отключить от электропитания оборудование, механизмы и т.п., обесточить помещение.
5. По возможности принять меры, по тушению пожара используя средства противопожарной защиты (углекислый огнетушитель).
6. По возможности предотвратить развитие аварии, обозначить место аварии [37].

6.4 Правовые и организационные вопросы обеспечения безопасности

Рабочая зона представляет собой все места, где работник должен находится или временно пребывать в процессе трудовой деятельности. Если

рабочее место неправильно организованно, то это может привести к получению производственной травмы, а также к неэффективности рабочего процесса. При организации рабочего места необходимо выполнять определенные требования эргономики [38]. К ним относятся:

- выбор положения работающего;
- пространственная компоновка рабочего места;
- размерные характеристики рабочего места;
- взаимное расположение рабочих мест;
- размещение технологической и организационной оснастки;

Также важен режим работы. При эффективном труде, необходимо выполнять следующее пункты:

- возможность перемены типов нагрузок и задач;
- наличие перерывов в работе: 5 минут через 1 час работы на дисплее. Во время перерыва следует производить физические упражнения с растяжением мышц спины и рук;
- возможность отдыха для глаз (гимнастика для глаз) [39].

Заключение

Алгоритмы принятого в 2013 году стандарта видеокодирования H.265/HEVC позволяют добиться почти двукратного повышения степени сжатия видеоданных по сравнению со стандартом предыдущего поколения H.264/AVC. Ценой такого роста эффективности является повышенная вычислительная сложность поиска оптимального режима кодирования. Одним из ключевых моментов, которым обусловлен рост вычислительных затрат, является возможность адаптивного разбиения кадра на блоки кодирования. С целью снижений времени, затрачиваемого на кодирование видеоизображений, в рамках данной работы разработаны быстрые алгоритмы адаптивного разбиения видеокадров на блоки кодирования в системе видеокодирования H.265/HEVC.

При выполнении диссертационной работы получены следующие результаты.

1. Проведен анализ методов и алгоритмов сжатия цифровых видеоданных вошедших в стандарт H.265/HEVC. Показано, что основной причиной повышенной вычислительной сложности процесса кодирования видеоизображений является перебор большого количества возможных вариантов кодирования, с целью поиска наиболее оптимального из них. Сделан вывод о том, что наиболее трудоёмким является перебор всех возможных вариантов разбиения кадра на блоки кодирования.
2. Проведен обзор известных критериев выбора размера блока кодирования, на основании которого выбран наиболее эффективный из них. Использование этого критерия позволяет на 48.81% сократить общее время, затрачиваемое вычислительной системой на кодирование видеоданных. Степень сжатия данных при этом снижается на 2.10%.

3. Показано, что процесс выбора оптимального разбиения блока кодирования на подблоки при применении выбранного алгоритма можно дополнительно ускорить, применив дополнительный критерий. Такая модификация позволяет дополнительно ускорить процесс кодирования при незначительном снижении степени сжатия.
4. С целью повышения эффективности предложенного метода выбора разбиений кадра на блоки кодирования разработаны варианты модификации используемых для ускорения алгоритмов. Показано, что при применении модификаций появляется возможность регулировать соотношения снижения степени сжатия и сокращения затрачиваемого на кодирование времени.
5. Разработана универсальная библиотека классов на языке программирования C++, реализующая предложенные алгоритмы быстрого разбиения кадра на блоки кодирования.
6. Разработана система кодирования видеоизображений в соответствии со стандартом H.265/HEVC, основанная на справочной реализации кодера, представленной разработчиками стандарта, и использующая разработанную библиотеку классов для принятия решений о разбиении кадра на блоки кодирования.
7. На разработанной системе кодирования проведены замеры эффективности разработанных алгоритмов. Показано, что предложенные алгоритмы позволяют достичь 51.94% сокращения времени кодирования при снижении степени сжатия на 2.39%. Оценка возможностей регулирования соотношения снижения степени сжатия и сокращения затрачиваемого на кодирование времени показала возможность настройки кодера таким образом, чтобы уменьшить рост битовой скорости вплоть до 1.06% при сокращении сохраняемого времени кодирования до 43.96%.

8. Разработанная библиотека классов, реализующая предложенные алгоритмы, внедрена в состав коммерческого продукта компании Элекард «Elecard Codec SDK».

Список литературы

1. Иванов Ю.А. Некоторые проблемы сжатия и передачи видео в реальном времени в беспроводных сетях // Электротехнические и информационные комплексы и системы. — 2009. — Т. 5. — № 1. — С. 62—64.
2. Grois, D. Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders / D. Grois, M. Marpe, A. Mulayoff, B. Itzhaky, O. Hadar // 30th Picture Coding Symposium 2013 (PCS 2013). — San Jose, CA, USA. — 8-11 Dec 2013. — P.1-4.
3. Jiang, W. Gradient based fast mode decision algorithm for intra prediction in HEVC / W. Jiang, H. Ma, Y. Chen // Consumer Electronics, Communications and Networks (CECNet), 2nd International Conference on. — April 2012. — P.1836-1840.
4. Kalali, E. A high performance and low energy intra prediction hardware for HEVC video decoding / E. Kalali, Y. Adibelli, I. Hamzaoglu // Conference on Design and Architectures for Signal and Image Processing (DASIP). — 23-25 October 2012. — P.1-8.
5. Khan, M.U.K. Hardware-software Collaborative Complexity Reduction Scheme for the Emerging HEVC Intra Encoder / M.U.K. Khan, M. Shafique, M. Grellert, J. Henkel // Design, Automation Test in Europe Conference Exhibition (DATE). — March 2013. — P.125-128.
6. Kim, Y. A Fast Intra-Prediction Method in HEVC Using Rate-Distortion Estimation Based on Hadamard Transform / Y. Kim, D. Jun, S.-H. Jung, J.S. Choi, J. Kim // ETRI Journal. — April 2013. — V.35. — Issue 2. — P.270-280.
7. Lambrecht, C.J. Perceptual Quality Measure using a SpatioTemporal Model of the Human Visual System / C.J. Lambrecht, O. Verscheure // SPIE. — San Jose, CA. — 1996. — 12 p.

8. Kim, J. Fast Coding Unit Size Decision Algorithm for Intra Coding in HEVC / J. Kim, Y. Choe, Y. Kim // IEEE International Conference on Consumer Electronics (ICCE). – January 2013. – P. 637–638.
9. Cho, S. Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding / S. Cho, M. Kim // IEEE Transactions on Circuits and Systems for Video Technology. – September 2013. – Vol. 23, № 9. – P. 1555–1564.
10. Shen, L. Fast CU Size Decision and Mode Decision Algorithm for HEVC Intra Coding. / L. Shen, Z. Zhang, P. An // IEEE Transaction on Consumer Electronics. – Vol. 59. – № 1. – February 2013. – P. 207–213.
11. Zhang, Y. Gradient-based Fast Decision for Intra Prediction in HEVC. / Y. Zhang, Z. Li, B. Li // IEEE Visual Communications and Image Processing (VCIP). – November 2012. – P. 1–6.
12. Min, B. A Fast CU Size Decision Algorithm for HEVC Intra Encoder / B. Min, R.C.C. Cheung // IEEE Transactions on Circuits and Systems for Video Technology. – May 2015. – Vol. 25, № 5. – P. 892–896.
13. Zhang, M. Entropy-Based Fast Largest Coding Unit Partition Algorithm in High-Efficiency Video Coding / M. Zhang, J. Qu, H. Bai // Entropy. – June 2013. – Vol. 15, № 6.– P. 2277–2287.
14. Пономарев О.Г. Критерий для быстрого выбора размера блока пространственного предсказания в системе видеокодирования HEVC / О.Г. Пономарев, М.П. Шарабайко, Д.Ю. Тё // Доклады ТУСУРа. 2015.– № 3. – С. 106–113.
15. Ponomarev O.G. Modified intra prediction unit size selection algorithm for H.265/HEVC compression system / Ponomarev O.G., Sharabayko M.P., Tyo D.Y., Strelnikov S.E. // Applied Mathematical Sciences. – 2015. – Vol. 9, no. 140. – P. 6985—6995.
16. Sullivan, G.J. Overview of the High Efficiency Video Coding (HEVC) Standard / G.J. Sullivan, J. Ohm, W.J. Han, T. Wiegand // Circuits and

Systems for Video Technology, IEEE Transactions on. — 2012. — V.22. — No.12. — P.1649-1668.

17. ГОСТ Р 52210-2004: Телевидение вещательное цифровое. Термины и определения.— М: ИПК Изд-во стандартов. — 2004. — 24 с.
18. Vanne J. Efficient mode decision schemes for HEVC inter prediction / J. Vanne, M. Viitanen, T.D. Hämäläinen // IEEE Transactions on Circuits and Systems for Video Technology. – September 2014. – Vol. 24, № 9. – P. 1579–1593.
19. Recommendation ITU-T H.265: High Efficiency Video coding, 2013 [Электронный ресурс]. – Режим доступа: https://www.itu.int/rec/T-REC-H.265-201304-S/en свободный (дата обращения: 27.04.2016).
20. Ohm, J.-R. Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC) / J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand // IEEE Transactions on Circuits and Systems for video Technology. – 2012. – Vol. 22. – № 12. – P. 1669–1684.
21. Ortega, A. Rate-Distortion Methods for Image and Video Compression / A. Ortega, K. Ramchandran // IEEE Signal Processing Magazine. – November, 1998. – P. 23–50.
22. Kim, I.-K. High Efficiency Video Coding (HEVC) Test Model 15 (HM15) Encoder Description / I.-K. Kim, K. McCann, K. Sugimoto, B. Bross, W.-J. Han, G. Sullivan // 17th Meeting: Valencia, ES. – 27 Mar. – 4 Apr. 2014.
23. Sullivan, G.J. Rate-distortion optimization for video compression. / G.J. Sullivan, T. Wiegand // IEEE Signal Process. Mag. 15(6), 74–90 (1998).
24. Wien, M High Efficiency Video Coding: Coding tools and specification // Springer-Verlag Berlin Heidelberg. – 2015.
25. Bjøntegaard G. Calculation of average PSNR differences between RD-curves // Technical Report VCEG-M33 – ITU-T SG16/Q6 – Austin, TX,

USA – April 2001 [Электронный ресурс]. – Режим доступа: http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc/ свободный (дата обращения: 24.08.2014).

26. Bossen F. Common Test Conditions and Software Reference Configurations // Document JCTVC-H1100. – JCT-VC, San Jose, CA – February 2012 [Электронный ресурс]. – Режим доступа: http://phenix.int-evry.fr/jct/ свободный (дата обращения: 24.08.2014).
27. H.265/HEVC Reference Software [Electronic Resource] // Fraunhofer. — URL: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/ (accessed: 27.04.2016).
28. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация. – М.: Информационно-издательский центр Минздрава России, 1974.
29. СНиП 2.2.4/2.1.8.562-96 Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки. – М.: Информационно-издательский центр Минздрава России, 1996.
30. СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений – М.: Информационно-издательский центр Минздрава России, 2003..
31. СанПиН 2.2.2/2.4.1340–03. Санитарно-эпидемиологические правила и нормативы «Гигиенические требования к персональным электронно- вычислительным машинам и организации работы».
32. .СанПиН 2.2.1/2.1.1.1278–03. Гигиенические требования к естественному, искусственному и совмещённому освещению жилых и общественных зданий.
33. ГОСТ Р 12.1.019-2009 ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты.
34. Федеральный классификационный каталог отходов [Электронный ресурс]. – 2013. – Режим доступа:

http://www.ecoguild.ru/faq/fedwastecatalog.htm, свободный. – Загл. с экрана.

35. Об утверждении правил обращения с отходами производства и потребления в части осветительных устройств, электрических ламп, ненадлежащие сбор, накопление, использование, обезвреживание, транспортирование и размещение которых может повлечь причинение вреда жизни, здоровью граждан, вреда животным, растениям и окружающей среде: Постановление Правительства Российской Федерации от 3 сентября 2010 года № 681.
36. ГОСТ Р 22.0.01-94. Безопасность в ЧС. Основные положения.
37. Федеральный закон Российской Федерации от 22 июля 2008 г. N 123-ФЗ "Технический регламент о требованиях пожарной безопасности".
38. ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования. – 1978
39. Эргономика рабочего места программиста. [Электронный ресурс] – Режим доступа: http://artelvyv.narod.ru/ergonom.htm.

Приложение А

(обязательное)

Раздел 3
Программная реализация быстрых алгоритмов видеосжатия (Software implementation of fast algorithms for video compression) Английский язык

Студент:

Группа	ФИО	Подпись	Дата
8БМ41	Тё Дмитрий Юрьевич		

Консультант кафедры ПМ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент каф. ПМ	Марченко В.В.	к.т.н		

Консультант – лингвист кафедры Иностранных языков Института кибернетики (ИЯИК):

Должность	ФИО	Ученая степень, звание	Подпись	Дата
зав. каф. ИЯИК	Сидоренко Т.В.	к.п.н		

4 Software implementation of fast algorithms for video compression

This section describes software implementation of a video compression system. Developed system is based on the open source implementation of HEVC video encoder produced by developers of the standard [27]. The main procedure of frame splitting mode selection in the basic implementation was modified using the algorithms proposed in section 3. Original algorithms were compiled as class-library in C++ programming language. This approach allowed to make the developed library unified and reusable.

4.1 Description of developed library

Developed algorithms are implemented in C++ programming language in a form of C++ class named `TEncSplitter`. Objects of this class may be used in video sequence encoding process on a CTU splitting phase to accelerate decision-making process, when CTU is split into quad-tree of CU's.

Public interface of the class `TEncSplitter` includes the next functions.

– `TEncSplitter()`

Description:

– `TEncSplitter()` is a constructor of the class. The function initializes all required variables, which are needed for a splitter-object to work.

– `Void init(TComYuv *pcYuvSrc, Int iDepth, Int iQP)`

Input parameters:

`pcYuvSrc` – a pointer to `TComYuv`. The variable contains the address of the picture area, which complies with the estimated coding unit.

`iDepth` – an integer value. The variable contains a value of the estimated coding unit depth in a quad-tree.

`iQP` – an nteger value. The variable contains a value of a quantization parameter *Qp*, which is applied to the estimated coding unit.

Description:

`init()` is a coding unit estimation initializer. The function prepares the information about the estimated coding unit. This information is needed to make decision about splitting or non-splitting mode of the coding unit.

– `Void decide()`

Description:

The function calculates the complexity characteristics described in section 3.2 to use a method proposed in [12] and decides if the estimated coding unit should be split or should not be split into four sub-units. After a decision process, the value of this decision is stored as enumerable C++ type `SplitterDecision` variable in the data member `m_decision` of the class.

– `SplitterDecision getDecision()`

Description:

The Function returns the value of data member `m_decision`, which contains a decision about necessity of splitting or non-splitting mode for the estimated coding unit.

– `Void setDepth(Int depth)`

Input parameters:

`Depth` – an integer value. The variable contains the new value of the coding unit depth in a quad-tree, which should be set for the splitter object.

Description:

The function sets the value of the coding unit depth in a quad-tree.

– `Void setQP(Int qp)`

Input parameters:

`qp` – an integer value. The variable contains the new value of the coding unit quantization parameter *Qp*.

Description:

The function sets the value of the coding unit quantization parameter *Qp*.

- `Int getDepth()`

  Description:

  The function returns the current value of the estimated coding unit depth in a quad-tree.

- `Int getQP()`

  Description:

  The aunction returns the current value of the quantization parameter *Qp* of the estimated coding unit

- `Double getThresholdRDC()`

  Description:

  The function returns the threshold values of *RDC* for the current coding unit depth in a quad-tree structure and the quantization parameter *Qp*.

Private functions of the class `TEncSplitter` are described below.

- `Void prepareGlbComplexities()`

  Description:

  The function calculates values of the global complexities for a decision-making process.

- `Void prepareLclComplexities()`

  Description:

  The function calculates values of the local complexities for the decision-making process.

- `Int getMeanY(Char filter)()`

  Input parameters:

`filter` – an integer value. The variable contains the index of a filter which should be used to calculate a mean value of pixels brightness:

«–1» – no filters;

«0» – horizontal filter;

«1» – vertical filter;

«2» – 45 degree diagonal filter;

«3» – 135 degree diagonal filter.

Default value «–1».

Description:

The function calculates and returns a mean value of a pixels brightness, which complies to the estimated coding unit. If it is necessary all pixels values are filtered by a filter corresponding to the index `filter` before the calculating process.

– `Int getMeanYSub(Int *vals, Char filter)()`

Input parameters:

`filter` – an integer value. The variable contains the index of a filter which should be used to calculate a mean value of pixels brightness:

«–1» – no filters;

«0» – horizontal filter;

«1» – vertical filter;

«2» – 45 degree diagonal filter;

«3» – 135 degree diagonal filter.

Default value «–1».

Выходные параметры:

`vals` – a pointer to integer array of four elements. The array contains mean values of four estimated coding units sub-units pixels.

Description:

The function calculates and returns mean values of four estimated coding units sub-units. If it is necessary before calculating process all pixels values are filtered by a filter corresponding to the index `filter`.

– `UChar getFilteredValue (Int x, Int y, Char filter)`

Input parameters:

`x`, `y` – integer values. Variables contain coordinates of the pixel to be filtered.

`filter` – an integer value. The variable contains the index of a filter which should be used to calculate a mean value of pixels brightness:

«–1» – no filters;

«0» – horizontal filter;

«1» – vertical filter;

«2» – 45 degree diagonal filter;

«3» – 135 degree diagonal filter.

Default value «–1».

Description:

The function calculates and returns filtered value of the estimated coding unit pixel with a location point (`x`, `y`).

– `static Int getThreasholdGlbForQP(Int qp, UInt depth)`

Input parameters:

`qp` – an integer value. The variable contains a quantization parameter *Qp*.

`depth` – an integer value. The variable contains a value of an estimated coding unit depth in quad-tree. Default value is «0».

Description:

The function calculates and returns a threshold value of a global complexity, corresponding to the given quantization parameter and the estimated coding unit depth in quad-tree.

– `static Int getThreasholdLclForQP(UInt depth)`

Input parameters:

– an integer value. The variable contains a value of an estimated coding unit depth in quad-tree. Default value is «0».

Description:

The function calculates and returns a threshold value of a local complexity, corresponding to the given estimated coding unit depth in quad-tree.

– `Int getMinGlobalComplexity()`

Description:

The function returns a minimal value of global complexities using all four directions.

– `Int getMaxGlobalComplexity()`

Description:

The function returns a maximal value of global complexities using all four directions.

– `Int getMinLocalComplexity()`

Description:

The function returns a minimal value of local complexities using all four directions.

– `Int getMaxLocalComplexity()`

Description:

The function returns a maximal value of local complexities using all four directions.

An additional structure required by `TEncSplitter` class is described next:

– `SplitterDecision` is an enumerable data type intended for containing of a decision if the coding unit should or should not be split. A variable of this type may contain one of next values:

- UNDETERMINED – an undetermined state means the *RDC* checking process is required to make a decision;

- SPLIT – a split state means the estimated coding unit should be split into four sub-units without any additional *RDC* checking process for the current coding unit depth;

- NO_SPLIT – a non-split state means a split process of the estimated coding unit should be stopped, the *RDC* checking process should be performed only for the current coding unit depth.

The source code of an interface and an implementation of the class described above is presented in annex Б.

To use developed library in an H.265/HEVC-compliant video compression system a corresponding header file must be included by `#include "TEncSplitter.h"` instruction. After that an object of `TEncSplitter` class should be instantiated. When a frame splitting process is performed the instantiated `TEncSplitter` object should be initialized by `init()` method with corresponding parameters. Then `decide()` function is called for the object to make a decision about necessity of a split or non-split mode for the estimated coding unit. After the decision is done, the corresponding value of the type `SplitterDicision` is available using the getter-function `getDecision()`. This function returns the `UNDETERMINED` value in case of impossibility of choosing one of splitting modes without the *RDC* estimation process. When *RDC*

is calculated the threshold value to stop splitting process may be accessed by the getter-function `getThresholdRDC()`.

4.2 Testing of developed library

A testing process of the developed library was performed based on reference encoding software HEVC Test Model version 15.0 (HM15) [22]. An efficiency estimation of the library performed by comparison of the original HM15 implementation with a modified one, which uses `TEncSplitter` library. As a measure of efficiency BD-rate and ΔT characteristics have been used (see section 3.1). During the testing process every measurement was performed on intra-only mode for video sequence with a length of one second.

Testing system properties are:

– CPU: Intel Core i5-3330 3.00 GHz;

– RAM: 8192 MB.

4.2.1 Building and configuring of the compression system

As a basic video encoding system the HM15 was used. This implementation of H.265/HEVC-complied encoder is presented by JCT-VC and available in a source code form [18]. Before estimating the efficiency of developed algorithms the HM15 building and debugging was performed. Its performance analysis and statistics collection were performed.

As a result of the HM15 encoding process analysis the fact that in frame splitting process the decision is made by an exhaustive search of all possible partitioning options may be concluded. For each coding unit the value of *RDC* is calculated using the formula (3). After that the estimation of sub-units *RDC* runs recursively. If the sum of sub-units *RDC* values is less than the one of the current coding unit, then the decision to split coding unit is made. Otherwise the coding

unit stays unite. Therefore, the system is applicable for the efficiency estimation of developed algorithms.

To estimate the efficiency of the developed algorithm a comparison of the estimated encoding system efficiency and the original algorithm of frame splitting statistics efficiency is performed. To collect that statistics the encoding process ran for the set of test sequences on the HM15 without any modifications. Measurements of a bitrate and an encoding time were performed for quantization parameter values of 22, 27, 32 and 37.

As a result of this section the basic encoding system implementation, that can be modified by developed algorithms of the fast optimal frame partitioning mode decision process, is configured and necessary statistics of the encoding time and the bitrate of output streams is collected.

4.2.2 Efficiency estimation of Min algorithm

Comparative measurements were performed to estimate the Min algorithm [12] efficiency. The results of the performance comparison of the Min criteria for the fast partition decision with the pure HM15 are presented in the table 7.

The first column of the table contains titles of the test video sequences. The second column depicts a picture resolution in pixels. The third and the fourth columns contain the values of BD-rate and ΔT obtained by the testing of the Min algorithm efficiency respectively.

Table 7 – Efficiency of the Min algorithm

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	1.73	47.07
PeopleOnStreet		1.59	44.87
SteamLocomotiveTrain		9.21	41.67
Kimono	1920×1080	6.90	64.55
ParkScene		1.26	50.19
Cactus		1.81	53.54
BQTerrace		1.03	46.55
BasketballDrive		2.10	62.55

BQMall	832×480	1.01	41.08
PartyScene		0.25	32.85
BasketballDrill		2.06	50.22
BQSquare	416×240	0.30	40.91
BlowingBubbles		1.03	37.89
BasketballPass		1.29	44.32
Vidyo1	1280×720	2.02	59.19
Vidyo3		5.16	54.61
Vidyo4		1.69	60.82
BasketballDrillText	832×480	1.11	48.44
ChinaSpeed	1024×768	1.48	53.72
SlideEditing	1280×720	0.29	38.57
SlideShow		0.75	51.44
В среднем		2.10	48.81

The table shows that usage of the Min algorithm allows reducing computing costs by 48.81% with 2.10% loss of compression ratio.

Results of the same algorithm presented in [12] show that authors were able to achieve 52.3% reduction of computing costs with 0.8% loss of the compression ratio. This discrepancy can be explained by different implementation of proposed algorithm. For example, authors of [12] don't describe the way of the local complexity calculation. To calculate them the filtered values of pixels, which lay in corresponded to the estimated coding unit picture area, are required. Those pixels use brightness values of the neighboring pixels to get filtered values. The pixels, which lay on a border of the estimated coding unit, need the pixels from neighboring coding units. Those pixels are not available in a common case. So it is impossible to calculate the filtered values for some pixels in the estimated coding unit. The implementation of the video compression system, presented in this paper, is based on an assumption that the edge pixels of the estimated coding unit are skipped whilst calculating of the local complexity values.

4.2.3 Efficiency estimation of the modified algorithm

To estimate the efficiency of the proposed algorithm of the fast split decisions a number of measurements was conducted. The results of these measurements are represented in tables 8–10. The first columns of the tables contain titles of the test sequences. The second columns show the resolution of the

corresponding video sequences. The third and the fours columns depict values of BD-rate and ΔT obtained by testing of the estimated modified algorithm efficiency respectively.

Table 8 – Efficiency of the combined Min and Kim algorithm

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	1.91	50.30
PeopleOnStreet		1.85	48.49
SteamLocomotiveTrain		9.23	42.82
Kimono	1920×1080	7.39	67.52
ParkScene		1.47	51.64
Cactus		2.16	54.34
BQTerrace		1.01	48.65
BasketballDrive		2.16	65.10
BQMall	832×480	1.03	43.19
PartyScene		0.29	32.69
BasketballDrill		1.85	50.62
BQSquare	416×240	0.31	40.37
BlowingBubbles		1.01	36.90
BasketballPass		1.28	46.54
Vidyo1	1280×720	3.15	65.23
Vidyo3		6.39	60.76
Vidyo4		2.55	67.26
BasketballDrillText	832×480	1.09	49.83
ChinaSpeed	1024×768	1.75	57.04
SlideEditing	1280×720	0.75	49.32
SlideShow		1.64	62.19
В среднем		2.39	51.94

In the table 8 the results of the speed increase and the compression ratio loss estimation for the combined algorithm of split decision described in the section 3.3 are represented. It is shown that this approach allows reducing computing costs by 51.92% with the loss of compression ratio of 2.39% on the average. In comparison with the results given in the section 4.2.1 the next conclusion can be made. Usage

Table 9 – Efficiency of the combined Min and Kim algorithm with modified threshold values of Min criteria

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	ΔT, %
Traffic	2560×1600	1.03	42.38
PeopleOnStreet		0.76	40.08
SteamLocomotiveTrain		2.37	52.94
Kimono	1920×1080	1.85	64.14
ParkScene		0.64	42.87

Cactus		0.96	46.56
BQTerrace		0.46	43.40
BasketballDrive		1.17	59.19
BQMall	832×480	1.12	35.01
PartyScene		0.17	24.13
BasketballDrill		1.64	40.90
BQSquare	416×240	0.28	30.20
BlowingBubbles		0.52	28.94
BasketballPass		0.53	38.70
Vidyo1	1280×720	1.30	40.03
Vidyo3		1.82	51.42
Vidyo4		0.86	39.16
BasketballDrillText	832×480	4.61	76.42
ChinaSpeed	1024×768	3.60	37.02
SlideEditing	1280×720	3.57	65.21
SlideShow		3.10	62.66
В среднем		1.54	45.78

Table 10 – Efficiency of combined Min and Kim algorithm with modified threshold values of both criteria

Название видеопоследовательности	Разрешение, в пикселях	BD-rate, %	Δ*T*, %
Traffic	2560×1600	0.66	39.67
PeopleOnStreet		0.49	37.57
SteamLocomotiveTrain		1.25	47.27
Kimono	1920×1080	1.33	58.50
ParkScene		0.53	42.35
Cactus		0.74	44.14
BQTerrace		0.36	40.67
BasketballDrive		0.64	54.58
BQMall	832×480	1.00	34.32
PartyScene		0.15	23.96
BasketballDrill		1.55	40.73
BQSquare	416×240	0.27	32.13
BlowingBubbles		0.51	30.78
BasketballPass		0.46	38.41
Vidyo1	1280×720	1.22	40.20
Vidyo3		1.32	48.51
Vidyo4		0.50	42.04
BasketballDrillText	832×480	2.10	73.12
ChinaSpeed	1024×768	3.50	35.89
SlideEditing	1280×720	2.02	60.77
SlideShow		1.61	57.56
В среднем		1.06	43.96

of the combined criteria of Min and Kim allows reaching higher speed increase with a small reduction of the compression ratio.

In the table 9 it is shown that the modification of the threshold values for Min criteria, which is based on splitting of the non-split and split threshold into two different values, allows reducing computing cost by 45.78% with the loss of compression ratio of 2.39% on the average. Comparing these values with the data

from the table 8 it is clear that this method significantly reduces the raise of the output stream bitrate. Nevertheless the encoding speed is also significantly reduces. In comparison with the statistical data presented in section 4.2.1 it is easy to see that the split threshold values of the Min criteria and in combine with the Kim algorithm it is possible to considerably reduce the encoded stream bitrate with a small loss in the encoding time.

The table 10 depicts results of the efficiency estimation in case when the threshold values are modified for both criteria. The reduction of computing cost, in comparison with the non-modified implementation of the encoder system HM15, is about 43.96%. The main goal of this modification is to get the minimal bitrate penalty, appearing in consequence of skipping of some partition modes, among which there are actual optimal modes. It means the most interesting measure here is BD-rate. In this case BD-rate is 1.06% on the average. This value has the most compression ratio among all conducted measures. Thereby the proposed modification allows reducing of the bitrate penalty almost twice in comparison with the original Min algorithm while the speed-up reduction is very small.

4.3 Main results and conclusion

The software implementation of the video compression system, based on the one presented by H.265/HEVC standard designers and utilizing the algorithms of fast splitting frame into coding units as proposed in the section 3, is described above. The results of the performance estimation of the developed system are shown. The main results of this section are next.

5. The universal class library, implementing proposed in the section 3 algorithms of the fast coding unit partitioning decision, is presented.
6. Debugging, configuring and modification of the reference implementation of the video compression system HM15 are done. As a

result of the modification the compression system, which uses developed class library on the frame partitioning stage, is obtained.

7. The efficiency estimations of the implemented video compression system, which uses different modification of proposed algorithm, are done. The efficiency estimation of the combined Min and Kim algorithm, described in the section 3.3, shows that the reduction of time, spent on encoding process, is 51.94% with the loss of compression ratio of 2.39% on the average. It is shown that as a result of the application of the threshold values modifications, proposed in the section 3.4, the combined algorithms compression ratio may be increased. In case of the split threshold values of the Min criteria the bitrate penalty is 1.54% versus 45.78% of the speed increase. If both criteria's thresholds are modified it is only 1.06% loss of bitrate versus 43.96% of the computing cost reduction.

The results of efficiency estimation of the developed system show that application of proposed fast partition algorithm allows encoding video sequences twice as fast as the non-modified encoding system. On the other hand, this approach reduces the compression ratio. The possibility of control a bitrate to computing cost ratio is shown. Finally, the fast H.265/HEVC compliant video compression system, which allows configuring the encoding process according to the specific requirements, is developed.

Приложение Б

Программный код интерфейса и реализации библиотеки классов

Листинг 1 – Содержание заголовочного файла TEncSplitter.h

```
#ifndef __TENCSPLT__
#define __TENCSPLT__

enum SplitterDecision {
	UNDETERMINED,
	SPLIT,
	NO_SPLIT
};

class TEncSplitter {
private:
	TComYuv* m_pcYuv;
	SplitterDecision m_decision;

	// complexities for Min's method
	Int m_iGh;
	Int m_iGv;
	Int m_iG45;
	Int m_iG135;
	Int m_iLh;
	Int m_iLv;
	Int m_iL45;
	Int m_iL135;
	Int m_iSubGh[4];  // four sub-units
	Int m_iSubGv[4];
	Int m_iSubG45[4];
	Int m_iSubG135[4];
	Int m_iMaxGsh;
	Int m_iMaxGsv;
	Int m_iMaxGs45;
	Int m_iMaxGs135;
	Int m_iMaxLsh;
	Int m_iMaxLsv;
	Int m_iMaxLs45;
	Int m_iMaxLs135;

	Int m_iMaxGh;           // max(G'h(k))
	Int m_iMaxGv;           // max(G'v(k))

	Int m_iQP;
	Int m_iDepth;

public:
	TEncSplitter();
	~TEncSplitter();

	Void init(TComYuv *pcYuvSrc, Int iDepth, Int iQP);
	Void decide();

	// getters/setters
	Int getQP() { return m_iQP; }
	Void setQP(Int qp) { m_iQP = qp; }
	Int getDepth() { return m_iDepth; }
```

```
        Void setDepth(Int depth) { m_iDepth = depth; }
        SplitterDecision getDecision() { return m_decision; }

        // threasholds for Kim's method for current instance
        Double getThreasholdRDC();

private:

        Void prepareGlbComplexities();
        Void prepareLclComplexities();

        Int getMeanY(Char filter = -1);
        Void getMeanYSub(Int *vals, Char filtered = -1);

        UChar getFilteredValue (Int x, Int y, Char filter = -1);

        // threasholds for Min's method
        static Int getThreasholdGlbForQP(Int qp, UInt depth = 0);
        static Int getThreasholdLclForQP(UInt depth = 0);

        Int getMinGlobalComplexity();
        Int getMaxGlobalComplexity();
        Int getMinLocalComplexity();
        Int getMaxLocalComplexity();


};

#endif //__TENCSPLT__
```

Листинг 2 – Содержание файла реализации TEncSplitter.cpp

```
#include "TEncSplitter.h"

#include <cmath>

using namespace std;

TEncSplitter::TEncSplitter ()
{
        m_pcYuv = NULL;
        m_decision = UNDETERMINED;
}

TEncSplitter::~TEncSplitter ()
{
}
Void TEncSplitter::init(TComYuv *pcYuvSrc, Int iDepth, Int iQP)
{
        m_pcYuv = pcYuvSrc;
        m_iDepth = iDepth;
        m_iQP = iQP;
}

Void TEncSplitter::decide()
{
        m_decision = UNDETERMINED;

        prepareGlbComplexities();
        prepareLclComplexities();
```

```
	Int iTHg = getThreasholdGlbForQP(m_iQP, m_iDepth);
	Int iTHl = getThreasholdLclForQP(m_iDepth);

	if (m_iGh <= iTHg && m_iLh <= iTHl)
	{
		if (m_iMaxGsh <= (iTHg >> 2) && m_iMaxGh <= (iTHg >> 2) &&
m_iMaxLsh <= (iTHl >> 2))
		{
			m_decision = NO_SPLIT;
		}
	}
	/*else */if (m_iGv <= iTHg && m_iLv <= iTHl)
	{
		if (m_iMaxGsv <= (iTHg >> 2) && m_iMaxGv <= (iTHg >> 2) &&
m_iMaxLsv <= (iTHl >> 2)) {
			m_decision = NO_SPLIT;
		}
	}
	/*else */if (m_iG45 <= iTHg && m_iL45 <= iTHl)
	{
		if (m_iMaxGs45 <= (iTHg >> 2) && m_iMaxLs45 <= (iTHl >> 2)) {
			m_decision = NO_SPLIT;
		}
	}
	/*else */if (m_iG135 <= iTHg && m_iL135 <= iTHl)
	{
		if (m_iMaxGs135 <= (iTHg >> 2) && m_iMaxLs135 <= (iTHl >> 2)) {
			m_decision = NO_SPLIT;
		}
	}
	/*else */if (min(min(min(m_iGh, m_iGv), m_iG45), m_iG135) > iTHg * 2.63
|| min(min(min(m_iLh, m_iLv), m_iL45), m_iL135) > iTHl * 2.63)
	{
		m_decision = SPLIT;
	}
	/*else
	{
		m_decision = UNDETERMINED;
	}*/
}

Void TEncSplitter::prepareGlbComplexities()
{
	m_iGh = 0;
	m_iGv = 0;
	m_iG45 = 0;
	m_iG135 = 0;

	// for current unit
	Int yMean = getMeanY();
	Int n = m_pcYuv->getWidth();
	for (Int i = 0; i < n; ++i) {
		for (Int j = 0; j < n; ++j) {
			Int difVal = abs(m_pcYuv->getLumaAddr()[j * n + i] - yMean);
			Int difValIdxFlip = abs(m_pcYuv->getLumaAddr()[i * n + j] -
yMean);

			if (j < (n >> 1))
			{
				m_iGh += difVal;
```

```
                        m_iGv += difValIdxFlip;
                    }
                    else if (j >= (n >> 1))
                    {
                        m_iGh -= difVal;
                        m_iGv -= difValIdxFlip;
                    }

                    if (j < i)
                    {
                        m_iG45 += difVal;
                    } else if (j > i) {
                        m_iG45 -= difVal;
                    }

                    if (j < n - i - 1) {
                        m_iG135 += difVal;
                    } else if (j > n - i - 1) {
                        m_iG135 -= difVal;
                    }
                }
            }

            m_iGh = abs(m_iGh);
            m_iGv = abs(m_iGv);
            m_iG45 = abs(m_iG45);
            m_iG135 = abs(m_iG135);

            // for sub-unit
            m_iSubGh[0] = 0;
            m_iSubGh[1] = 0;
            m_iSubGh[2] = 0;
            m_iSubGh[3] = 0;
            m_iSubGv[0] = 0;
            m_iSubGv[1] = 0;
            m_iSubGv[2] = 0;
            m_iSubGv[3] = 0;
            m_iSubG45[0] = 0;
            m_iSubG45[1] = 0;
            m_iSubG45[2] = 0;
            m_iSubG45[3] = 0;
            m_iSubG135[0] = 0;
            m_iSubG135[1] = 0;
            m_iSubG135[2] = 0;
            m_iSubG135[3] = 0;
            Int yMeanSub[4];
            getMeanYSub(yMeanSub);
            Int ns = n >> 1;
            for (Int i = 0; i < ns; ++i)
            {
                for (Int j = 0; j < ns; ++j)
                {
                    Int difVal0 = abs(m_pcYuv->getLumaAddr()[j * n + i] -
yMeanSub[0]);
                    Int difValIdxFlip0 = abs(m_pcYuv->getLumaAddr()[i * n + j] -
yMeanSub[0]);
                    Int difVal1 = abs(m_pcYuv->getLumaAddr()[j * n + i + ns] -
yMeanSub[1]);
                    Int difValIdxFlip1 = abs(m_pcYuv->getLumaAddr()[i * n + j +
ns] - yMeanSub[1]);
```

```
                Int difVal2 = abs(m_pcYuv->getLumaAddr()[(ns + j) * n + i] -
yMeanSub[2]);
                Int difValIdxFlip2 = abs(m_pcYuv->getLumaAddr()[(ns + i) * n
+ j] - yMeanSub[2]);
                Int difVal3 = abs(m_pcYuv->getLumaAddr()[(ns + j) * n + i +
ns] - yMeanSub[3]);
                Int difValIdxFlip3 = abs(m_pcYuv->getLumaAddr()[(ns + i) * n
+ j + ns] - yMeanSub[3]);

                if (j < (ns >> 1))
                {
                      m_iSubGh[0] += difVal0;
                      m_iSubGv[0] += difValIdxFlip0;
                      m_iSubGh[1] += difVal1;
                      m_iSubGv[1] += difValIdxFlip1;
                      m_iSubGh[2] += difVal2;
                      m_iSubGv[2] += difValIdxFlip2;
                      m_iSubGh[3] += difVal3;
                      m_iSubGv[3] += difValIdxFlip3;
                }
                else if (j >= (ns >> 1))
                {
                      m_iSubGh[0] -= difVal0;
                      m_iSubGv[0] -= difValIdxFlip0;
                      m_iSubGh[1] -= difVal1;
                      m_iSubGv[1] -= difValIdxFlip1;
                      m_iSubGh[2] -= difVal2;
                      m_iSubGv[2] -= difValIdxFlip2;
                      m_iSubGh[3] -= difVal3;
                      m_iSubGv[3] -= difValIdxFlip3;
                }

                if (j < i)
                {
                      m_iSubG45[0] += difVal0;
                      m_iSubG45[1] += difVal1;
                      m_iSubG45[2] += difVal2;
                      m_iSubG45[3] += difVal3;
                }
                else if (j > i)
                {
                      m_iSubG45[0] -= difVal0;
                      m_iSubG45[1] -= difVal1;
                      m_iSubG45[2] -= difVal2;
                      m_iSubG45[3] -= difVal3;
                }

                if (j < ns - i - 1) {
                      m_iSubG135[0] += difVal0;
                      m_iSubG135[1] += difVal1;
                      m_iSubG135[2] += difVal2;
                      m_iSubG135[3] += difVal3;
                }
                else if (j > ns - i - 1)
                {
                      m_iSubG135[0] -= difVal0;
                      m_iSubG135[1] -= difVal1;
                      m_iSubG135[2] -= difVal2;
                      m_iSubG135[3] -= difVal3;
                }
          }
```

```
	}

	m_iMaxGsh = 0;
	m_iMaxGsv = 0;
	m_iMaxGs45 = 0;
	m_iMaxGs135 = 0;
	for (Int i = 0; i < 4; ++i)
	{
		m_iMaxGsh = max(m_iMaxGsh, abs(m_iSubGh[i]));
		m_iMaxGsv = max(m_iMaxGsv, abs(m_iSubGv[i]));
		m_iMaxGs45 = max(m_iMaxGs45, abs(m_iSubG45[i]));
		m_iMaxGs135 = max(m_iMaxGs135, abs(m_iSubG135[i]));
	}


	// G'h(k), G'v(k)
	Int iGhk1 = 0;
	Int iGhk1Mean = 0;
	Int iGhk2 = 0;
	Int iGhk2Mean = 0;
	Int iGhk3 = 0;
	Int iGhk3Mean = 0;
	Int iGhk4 = 0;
	Int iGhk4Mean = 0;

	Int iGvk1 = 0;
	Int iGvk1Mean = 0;
	Int iGvk2 = 0;
	Int iGvk2Mean = 0;
	Int iGvk3 = 0;
	Int iGvk3Mean = 0;
	Int iGvk4 = 0;
	Int iGvk4Mean = 0;

	// среднее считаем
	for (Int i = 0; i < (n >> 2); ++i)
	{
		for (Int j = 0; j < (n >> 1); ++j)
		{
			iGhk1Mean += m_pcYuv->getLumaAddr()[j * n + i];
	// верхняя половинка
			iGhk1Mean += m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i];
	// нижняя половинка
			iGhk2Mean += m_pcYuv->getLumaAddr()[j * n + i + (n >> 2)];
			iGhk2Mean += m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i +
(n >> 2)];
			iGhk3Mean += m_pcYuv->getLumaAddr()[j * n + i + (n >> 2) *
2];
			iGhk3Mean += m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i +
(n >> 2) * 2];
			iGhk4Mean += m_pcYuv->getLumaAddr()[j * n + i + (n >> 2) *
3];
			iGhk4Mean += m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i +
(n >> 2) * 3];

			iGvk1Mean += m_pcYuv->getLumaAddr()[i * n + j];
			// левая половинка
			iGvk1Mean += m_pcYuv->getLumaAddr()[i * n + j + (n >> 1)];
			// правая половинка
			iGvk2Mean += m_pcYuv->getLumaAddr()[(i + (n >> 2)) * n + j];
```

```
                    iGvk2Mean += m_pcYuv->getLumaAddr()[(i + (n >> 2)) * n + j +
(n >> 1)];
                    iGvk3Mean += m_pcYuv->getLumaAddr()[(i + (n >> 2) * 2) * n +
j];
                    iGvk3Mean += m_pcYuv->getLumaAddr()[(i + (n >> 2) * 2) * n +
j + (n >> 1)];
                    iGvk4Mean += m_pcYuv->getLumaAddr()[(i + (n >> 2) * 3) * n +
j];
                    iGvk4Mean += m_pcYuv->getLumaAddr()[(i + (n >> 2) * 3) * n +
j + (n >> 1)];
              }
        }

        iGhk1Mean /= n * (n >> 2);
        iGhk2Mean /= n * (n >> 2);
        iGhk3Mean /= n * (n >> 2);
        iGhk4Mean /= n * (n >> 2);

        iGvk1Mean /= n * (n >> 2);
        iGvk2Mean /= n * (n >> 2);
        iGvk3Mean /= n * (n >> 2);
        iGvk4Mean /= n * (n >> 2);

        auto CalcV = [&](int x, int y, int mean, int offset_x, int offset_y)
        {
              return abs(m_pcYuv->getLumaAddr()[(y + offset_y) * n + x] - mean)
- abs(m_pcYuv->getLumaAddr()[(y + offset_y) * n + x + offset_x] - mean);
        };

        auto CalcH = [&](int x, int y, int mean, int offset_x, int offset_y)
        {
              return CalcV(y, x, mean, offset_y, offset_x);
        };

        for (Int i = 0; i < (n >> 2); ++i)
        {
              for (Int j = 0; j < (n >> 1); ++j)
              {
                    iGhk1 += abs(m_pcYuv->getLumaAddr()[j * n + i] - iGhk1Mean)
- abs(m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i] - iGhk1Mean);
                    iGhk2 += abs(m_pcYuv->getLumaAddr()[j * n + i + (n >> 2)] -
iGhk2Mean) - abs(m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i + (n >> 2)] -
iGhk2Mean);
                    iGhk3 += abs(m_pcYuv->getLumaAddr()[j * n + i + (n >> 1)] -
iGhk3Mean) - abs(m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i + (n >> 1)] -
iGhk3Mean);
                    iGhk4 += abs(m_pcYuv->getLumaAddr()[j * n + i + (n >> 2) *
3] - iGhk4Mean) - abs(m_pcYuv->getLumaAddr()[(j + (n >> 1)) * n + i + (n >>
2) * 3] - iGhk4Mean);
                    iGvk1 += CalcV(j, i, iGvk1Mean, n / 2, 0 * n / 4);
                    iGvk2 += CalcV(j, i, iGvk2Mean, n / 2, 1 * n / 4);
                    iGvk3 += CalcV(j, i, iGvk3Mean, n / 2, 2 * n / 4);
                    iGvk4 += CalcV(j, i, iGvk4Mean, n / 2, 3 * n / 4);

                    const int a = abs(m_pcYuv->getLumaAddr()[i * n + j] -
iGvk1Mean) - abs(m_pcYuv->getLumaAddr()[i * n + j + (n >> 1)] - iGvk1Mean);

                    assert(CalcV(j, i, iGvk1Mean, n / 2, 0 * n / 4) ==
abs(m_pcYuv->getLumaAddr()[i * n + j] - iGvk1Mean) - abs(m_pcYuv-
>getLumaAddr()[i * n + j + (n >> 1)] - iGvk1Mean));
```

```
                  assert(CalcV(j, i, iGvk2Mean, n / 2, 1 * n / 4) ==
abs(m_pcYuv->getLumaAddr()[(i + (n >> 2)) * n + j] - iGvk2Mean) -
abs(m_pcYuv->getLumaAddr()[(i + (n >> 2)) * n + j + (n >> 1)] - iGvk2Mean));
                  assert(CalcV(j, i, iGvk3Mean, n / 2, 2 * n / 4) ==
abs(m_pcYuv->getLumaAddr()[(i + (n >> 1)) * n + j] - iGvk3Mean) -
abs(m_pcYuv->getLumaAddr()[(i + (n >> 1)) * n + j + (n >> 1)] - iGvk3Mean));
                  assert(CalcV(j, i, iGvk4Mean, n / 2, 3 * n / 4) ==
abs(m_pcYuv->getLumaAddr()[(i + (n >> 2) * 3) * n + j] - iGvk4Mean) -
abs(m_pcYuv->getLumaAddr()[(i + (n >> 2) * 3) * n + j + (n >> 1)] -
iGvk4Mean));
            }
      }

      iGhk1 = abs(iGhk1);
      iGhk2 = abs(iGhk2);
      iGhk3 = abs(iGhk3);
      iGhk4 = abs(iGhk4);

      iGvk1 = abs(iGvk1);
      iGvk2 = abs(iGvk2);
      iGvk3 = abs(iGvk3);
      iGvk4 = abs(iGvk4);

      m_iMaxGh = max(max(max(iGhk1, iGhk2), iGhk3), iGhk4);
      m_iMaxGv = max(max(max(iGvk1, iGvk2), iGvk3), iGvk4);
}

Void TEncSplitter::prepareLclComplexities()
{
      m_iLh = 0;
      m_iLv = 0;
      m_iL45 = 0;
      m_iL135 = 0;

      Int meanVals[4] = {0, 0, 0, 0};           // for 4 directions (hor,
vert, 45, 135)
      for (Int i = 0; i < 4; ++i)
      {
            meanVals[i] = getMeanY(i);
      }

      Int n = m_pcYuv->getWidth() - 1;

      for (Int i = 1; i < n; ++i)
      {
            for (Int j = 1; j < n; ++j)
            {
                  m_iLh += abs(getFilteredValue(i, j, 0) - meanVals[0]);
                  m_iLv += abs(getFilteredValue(i, j, 1) - meanVals[1]);
                  m_iL45 += abs(getFilteredValue(i, j, 2) - meanVals[2]);
                  m_iL135 += abs(getFilteredValue(i, j, 3) - meanVals[3]);
            }
      }

      Int n2 = m_pcYuv->getWidth();
      n = n2 >> 1;
      for (Int filter = 0; filter < 4; ++filter)
      {
            Int meanValsSub[4] = {0, 0, 0, 0};  // four sub-units for current
filter
            getMeanYSub(meanValsSub, filter);
```

```
            Int *curMaxLs = NULL;
            switch (filter)
            {
            case 0:
                  curMaxLs = &m_iMaxLsh;
                  break;

            case 1:
                  curMaxLs = &m_iMaxLsv;
                  break;

            case 2:
                  curMaxLs = &m_iMaxLs45;
                  break;

            case 3:
                  curMaxLs = &m_iMaxLs135;
                  break;

            default:
                  break;
            }

            assert(curMaxLs);

            (*curMaxLs) = 0;

            Int curVals[4] = {0, 0, 0, 0};

            for (Int y = 0; y < n; ++y)
            {
                  for (Int x = 0; x < n; ++x)
                  {
                        if (y && x)
                        {
                              curVals[0] += abs(getFilteredValue(x, y, filter)
- meanValsSub[0]);
                        }
                        if (y && x + n + 1 < n2)
                        {
                              curVals[1] += abs(getFilteredValue(x + n, y,
filter) - meanValsSub[1]);
                        }
                        if (n + y + 1 < n2 && x)
                        {
                              curVals[2] += abs(getFilteredValue(x, (n + y),
filter) - meanValsSub[2]);
                        }
                        if (n + x + 1 < n2 && y + n + 1 < n2)
                        {
                              curVals[3] += abs(getFilteredValue(x + n, n + y,
filter) - meanValsSub[3]);
                        }
                  }
            }

            for (Int i = 0; i < 4; ++i) {
                  (*curMaxLs) = max(*curMaxLs, curVals[i]);
            }
      }
```

```
}

/**
 * @param filter -1 - no filter, 0 - hor, 1 - vert, 2 - 45, 3 - 135
 */
Int TEncSplitter::getMeanY(Char filter)
{
	Int res = 0;

	if (filter > -1)
	{
		Int n = m_pcYuv->getWidth() - 1;

		for (Int i = 1; i < n; ++i)
		{
			for (Int j = 1; j < n; ++j)
			{
				res += getFilteredValue(i, j, filter);
			}
		}
		--n;
		res /= n * n;
	}
	else
	{
		Int n = m_pcYuv->getWidth();

		for (Int i = 0; i < n; ++i)
		{
			for (Int j = 0; j < n; ++j)
			{
				res += m_pcYuv->getLumaAddr()[i * n + j];
			}
		}
		res /= n * n;
	}

	return res;
}

/**
 * @param filter -1 - no filter, 0 - hor, 1 - vert, 2 - 45, 3 - 135
 */
UChar TEncSplitter::getFilteredValue (Int x, Int y, Char filter)
{
	Int n = m_pcYuv->getWidth();
	UChar res = m_pcYuv->getLumaAddr()[y * n + x];

	switch (filter)
	{
	case 0:
		res = m_pcYuv->getLumaAddr()[y * n + x - 1] - m_pcYuv-
>getLumaAddr()[y * n + x + 1];
		break;

	case 1:
		res = m_pcYuv->getLumaAddr()[(y - 1) * n + x] - m_pcYuv-
>getLumaAddr()[(y + 1) * n + x];
		break;

	case 2:
```

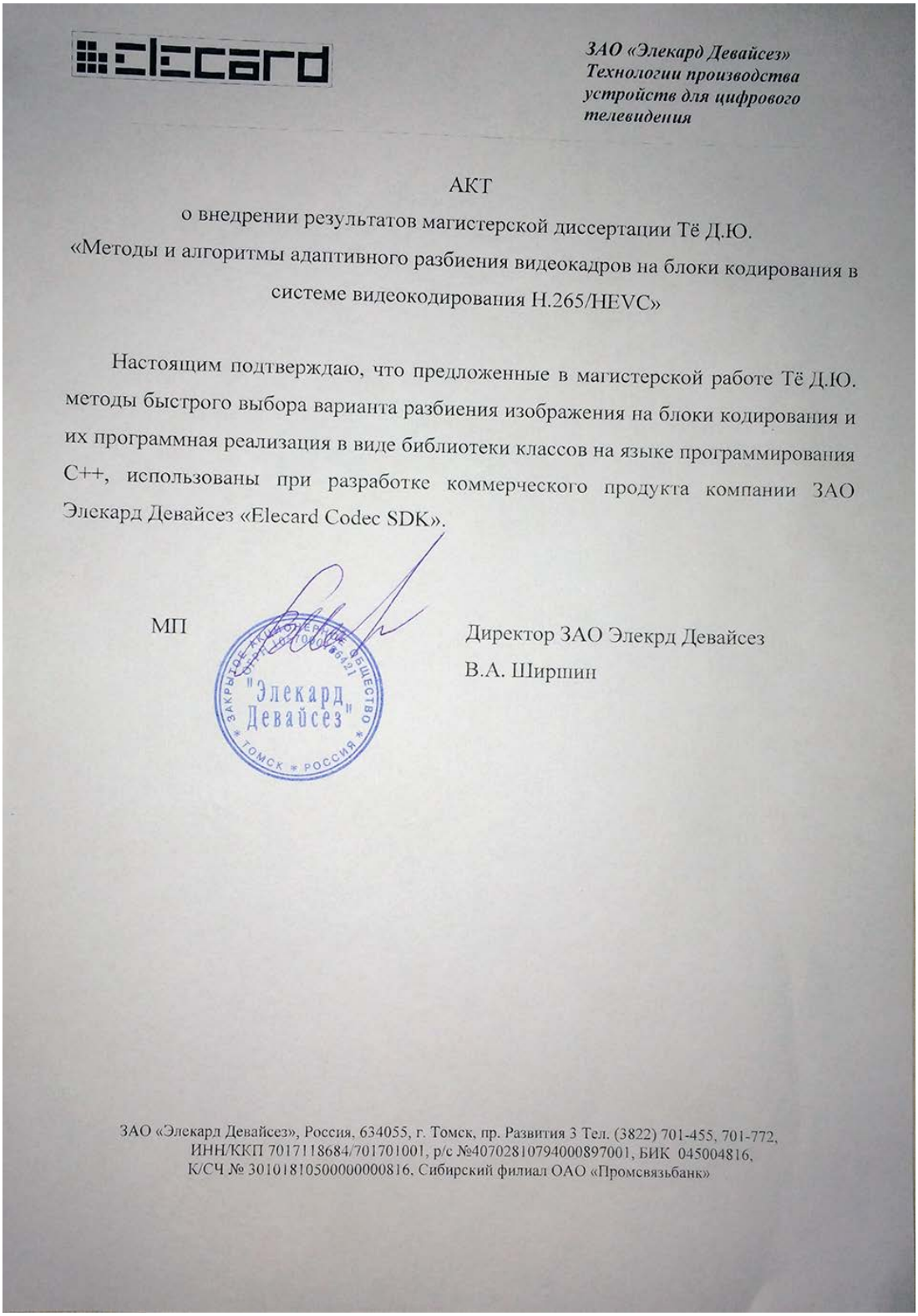
```
            res = m_pcYuv->getLumaAddr()[(y - 1) * n + x - 1] - m_pcYuv-
>getLumaAddr()[(y + 1) * n + x + 1];
            break;

      case 3:
            res = m_pcYuv->getLumaAddr()[(y - 1) * n + x + 1] - m_pcYuv-
>getLumaAddr()[(y + 1) * n + x - 1];
            break;

      default:
            break;
      }

      return res;
}

Void TEncSplitter::getMeanYSub(Int* vals, Char filter)
{
      Int n2 = m_pcYuv->getWidth();
      Int n = n2 >> 1;

      memset(vals, 0, (sizeof(int)) << 2);
      if (filter == -1)
      {
            for (Int i = 0; i < n; ++i)
            {
                  for (Int j = 0; j < n; ++j)
                  {
                        vals[0] += m_pcYuv->getLumaAddr()[i * n2 + j];
                        vals[1] += m_pcYuv->getLumaAddr()[i * n2 + j + n];
                        vals[2] += m_pcYuv->getLumaAddr()[(n + i) * n2 + j];
                        vals[3] += m_pcYuv->getLumaAddr()[(n + i) * n2 + j +
n];
                  }
            }

            vals[0] /= n * n;
            vals[1] /= n * n;
            vals[2] /= n * n;
            vals[3] /= n * n;
      }
      else
      {
            for (Int i = 0; i < n; ++i)
            {
                  for (Int j = 0; j < n; ++j)
                  {
                        if (i && j)
                        {
                              vals[0] += getFilteredValue(i, j, filter);
                        }
                        if (j && i + n + 1 < n2)
                        {
                              vals[1] += getFilteredValue(i + n, j, filter);
                        }
                        if (n + j + 1 < n2 && i)
                        {
                              vals[2] += getFilteredValue(i, (n + j), filter);
                        }
                        if (n + j + 1 < n2 && i + n + 1 < n2)
                        {
```

```
                                vals[3] += getFilteredValue(i + n, n + j,
filter);
                        }
                    }
                }

                vals[0] /= n * n - ((n << 1) - 1);
                vals[1] /= n * n - (n << 1) + 1;
                vals[2] /= n * n - (n << 1) + 1;
                vals[3] /= n * n - (n << 1) + 1;
        }
}

Int TEncSplitter::getThreasholdGlbForQP(Int qp, UInt depth)
{
        float res = 0;

        switch(qp)
        {
        case 22:
                res = 448.;
                break;

        case 27:
                res = 704.;
                break;

        case 32:
                res = 832.;
                break;

        case 37:
                res = 1216.;
                break;

        default:
                break;
        }

        for (Int i = 0; i < depth; ++i)
        {
                res *= .75;
        }

        return (Int) res;
}

Int TEncSplitter::getThreasholdLclForQP(UInt depth)
{
        float res = 5120.;

        for (Int i = 0; i < depth; ++i)
        {
                res *= .75;
        }

        return (Int) res;
}

Int TEncSplitter::getMinGlobalComplexity()
{
```

```
      return min(min(min(m_iGh, m_iGv), m_iG45), m_iG135);
}

Int TEncSplitter::getMaxGlobalComplexity()
{
      return max(max(max(m_iGh, m_iGv), m_iG45), m_iG135);
}

Int TEncSplitter::getMinLocalComplexity()
{
      return min(min(min(m_iLh, m_iLv), m_iL45), m_iL135);
}

Int TEncSplitter::getMaxLocalComplexity()
{
      return max(max(max(m_iLh, m_iLv), m_iL45), m_iL135);
}

Double TEncSplitter::getThreasholdRDC()
{
      Double res = 0;

      switch (m_iDepth)
      {
      case 0:
            res = 1265 * exp(0.086 * m_iQP);
            break;

      case 1:
            res = 179.5 * exp(0.1329 * m_iQP);
            break;

      case 2:
            res = 26.41 * exp(0.1678 * m_iQP);
            break;

      case 3:
            res = 1.054 * exp(0.254 * m_iQP);
            break;

      default:
            break;
      }

      return res;
}
```

Приложение В

Акт о внедрении



ЗАО «Элекард Девайсез»
Технологии производства устройств для цифрового телевидения

АКТ

о внедрении результатов магистерской диссертации Тё Д.Ю.
«Методы и алгоритмы адаптивного разбиения видеокадров на блоки кодирования в системе видеокодирования H.265/HEVC»

Настоящим подтверждаю, что предложенные в магистерской работе Тё Д.Ю. методы быстрого выбора варианта разбиения изображения на блоки кодирования и их программная реализация в виде библиотеки классов на языке программирования C++, использованы при разработке коммерческого продукта компании ЗАО Элекард Девайсез «Elecard Codec SDK».

МП

Директор ЗАО Элекрд Девайсез
В.А. Ширшин

ЗАО «Элекард Девайсез», Россия, 634055, г. Томск, пр. Развития 3 Тел. (3822) 701-455, 701-772,
ИНН/ККП 7017118684/701701001, р/с №40702810794000897001, БИК 045004816,
К/СЧ № 30101810500000000816, Сибирский филиал ОАО «Промсвязьбанк»