УДК 629.391:004.627

Д.Э. Двухглавов[1], Н.Н. Николаев[1], В.В. Твердохлеб[2]

[1] *Национальный технический университет «ХПИ», Харьков*
[2] *Национальный технический университет радиоэлектроники, Харьков*

ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ КОМПРЕССИИ НА ОСНОВЕ БИТОВОГО ПРЕДСТАВЛЕНИЯ БЛОКОВ ВИДЕОКАДРОВ

Представленные результаты являются этапом исследований, направленных на уменьшение битовой скорости обработки кадров видеопотока. Выполняется проектирование программного компонента, который позволит формировать битовые кубы из исходного представления блока видеокадра, а затем битовую последовательность для передачи блока, на основе выбранного байта передачи, а также выполнять обратное преобразование. Описаны наиболее значимые для разработки данного компонента UML-диаграммы, определенные соответствующей спецификацией в парадигме объектного подхода к проектированию программного обеспечения.

Ключевые слова: *компрессия видеокадров, битовое представление блоков видеокадра, объектное проектирование программного обеспечения, UML-диаграммы.*

Введение

Актуальность управления компрессией видеопотока обуславливается возрастающими требованиями к качеству предоставления услуг видеосервиса пользователям мобильных устройств, количество которых постоянно растет. На сегодняшней день показано [1, 2], что эффективное решение задачи обеспечения непрерывности передачи видеоинформации с заданным качеством связано с разработкой методов, которые предполагают управление объемом информации на источнике видеоданных. В [1, 3, 4] предложены подходы к управлению параметрами обработки для различных вариантов представления кадров видеопотока.

Метод, изложенный в источнике [4], предполагает представление блоков видеокадра в трехмерном пространстве.

Трансформация исходного изображения в битовый куб открывает возможность использовать различные варианты частичной передачи изображения, которые позволят снизить объем передаваемой информации.

Такие перспективы определяют целесообразность разработки программного обеспечения, которое позволит исследовать возможность применения такого представления в качестве основы для построения алгоритмов управления битовой скоростью компрессии видеокадров.

Целью статьи является проектирование программного компонента, позволяющего формировать битовую последовательность для передачи блоков видеокадра, представленных в виде битового куба, и ее восстановление на основе выбранного управляющего байта передачи, определяющего степень компрессии.

Основная часть

Для проектирования программного обеспечения используется объектный подход, который предполагает представление различных аспектов будущего программного обеспечения с использованием UML-диаграмм. Применение объектного подхода позволяет разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы, поэтому ООП является одним из наиболее интенсивно развивающихся направлений теоретического и прикладного программирования [5].

Спецификация разрабатываемого программного обеспечения при использовании UML объединяет несколько моделей: использования, логическую, реализации, процессов и развертывания.

Модель использования программного компонента предполагает, что пользователь загружает исходное изображение в формате bmp с диска и после обработки (компрессии) получает последовательность битов для передачи. Данная последовательность сразу поступает на обратное преобразование (декомпрессию), и восстановленное изображение записывается на диск.

Модель реализации программной компоненты предполагает создание единственного файла для запуска программы. Вследствие этого и модель размещения является простой, так как предполагается однопользовательский режим запуска компонента, а использование баз данных и удаленный доступ в данной компоненте не предполагается.

Разрабатываемый программный компонент будет разбивать исходное изображение на блоки (8x8), каждый из которых будет подвергнут дальнейшему преобразованию, в соответствии со схемой, которая

представлена на рис. 1. Результаты обработки блоков будут преобразованы в единое изображение.

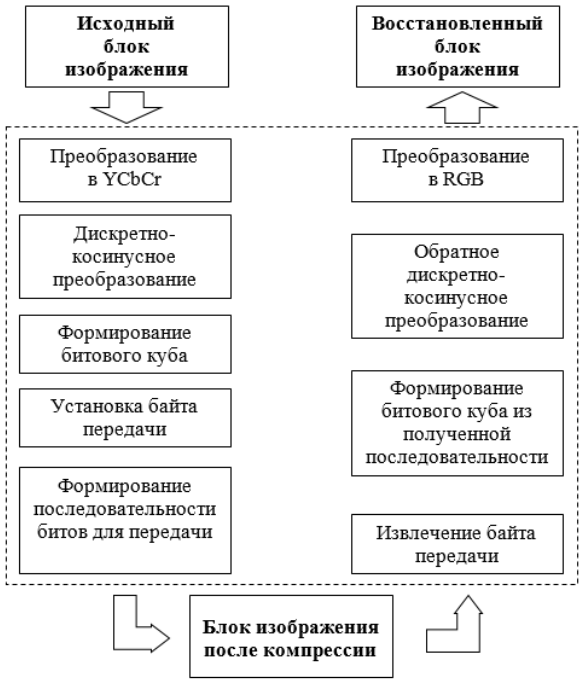

Рис. 1. Схема прямого и обратного преобразования блока видеокадра

Логическую модель в данной задаче образуют такие объекты как «изображение», «блок изображения», «последовательность передачи».

Для представления изображения в целом вводится класс TransferImage со следующими полями:

Width (int) – ширина изображения в пикселах;

Height (int) – высота изображения в пикселах;

BlockWidth (int) – ширина блока;

BlockHeight (int) – высота блока.

Предлагаемые наименования типов данных представлены для реализации компонента на языке программирования c++.

Для обработки изображения необходимо реализовать следующие методы:

void LoadFromFile(FileName) – процедура загрузки изображения из файла FileName;

void SaveToFile(FileName) - процедура сохранения изображения на жесткий диск;

void BlocksImage(BlockSize) – процедура разбиения на блоки размером BlockSize x BlockSize.

Для проектирования класса BlockOfImage следует рассмотреть более детально последовательность преобразования исходного блока видеокадра в представление в трехмерном пространстве.

Получение такого представления начинается с преобразования блоков видеокадра из модели RGB в модель YCbCr, после чего к ним применяется DC-преобразование. Математические выражения для такого преобразования представлены, например, в [6]. В результате для отдельного блока будет получена матрица размером 8x8, которую будем обозначать Y, $Y = \left\| Y_{ij} \right\|$, где Y_{ij} – это значение элемента (i, j) трансформанты, полученной после DC-преобразования. Экспериментальные исследования на различных типах изображений показали, что в диапазон значений [-128;127] попадают все значений матриц, кроме некоторых значений элементов (0;0) матриц («нулевых» элементов). Чтобы обрабатывать изображение далее по единому алгоритму можно старшие разряды двоичного представления вынести в отдельные переменные. Тогда каждый элемент трансформанты можно представить в виде столбца битов, соответствующих представлению соответствующего элемента в двоичном виде. Фактически выполняется следующее преобразование:

$$\left\| Y_{ij} \right\| \to \left\| \left\langle y_{ij7}, y_{ij6}, y_{ij5}, y_{ij4}, y_{ij3}, y_{ij2}, y_{ij1}, y_{ij0} \right\rangle^{T} \right\|,$$

$$y_{ijm} \in \{0,1\},\ i = \overline{0,7};\ j = \overline{0,7};\ m = \overline{7,0},$$

где y_{ijm} - $m-$й бит двоичного разложения элемента (i, j) или элемент (i, j) $m-$го слоя битового куба.

Объединение двоичных представлений всех элементов матрицы Y составит битовый куб. При этом верхний слой данного куба образуют старшие биты двоичного представления. Принятое расположение слоев битового куба предложено с учетом способа организации передачи блоков видеокадра.

Таким образом, в классе BlockOfImage следует предусмотреть:

PX[BlockWidth, BlockHeight] (int) – массив координат точек в исходном изображении;

R[BlockWidth, BlockHeight] (sbyte);

G[BlockWidth, BlockHeight] (sbyte);

B[BlockWidth, BlockHeight] (sbyte) – массивы значений компонент R, G, B точек блока исходного изображения;

Y[BlockWidth, BlockHeight] (byte);

Cb[BlockWidth, BlockHeight] (byte);

Cr[BlockWidth, BlockHeight] (byte) – массивы значений компонент Y, Cb, Cr после преобразования из RGB точек блока исходного изображения;

DCTY[BlockWidth, BlockHeight] (byte);

DCTCb[BlockWidth, BlockHeight] (byte);

DCTCr[BlockWidth, BlockHeight] (byte) – массивы значений компонент Y, Cb, Cr точек блока после DCT;

HY (byte);

HCb (byte);

HCr (byte) -– «старшие» байты «нулевых» элементов соответствующих компонент после DCT;

BitCubeY[8][8] (byte);

BitCubeCb[8][8] (byte);

BitCubeCr[8][8] (byte) – битовые кубы соответствующих компонент трансформанты до передачи;

TB (byte) – байт передачи для блока;

BBitCubeY[8][8] (byte);

BBitCubeCb[8][8] (byte);

BBitCubeCr[8][8] (byte); – битовые кубы соответствующих компонент трансформанты после приема и декомпрессии;

BHY (byte);

BHCb (byte);

BHCr (byte) – «старшие» байты «нулевых» элементов соответствующих компонент после передачи;

BDCTY[BlockWidth, BlockHeight] (byte);

BDCTCb[BlockWidth, BlockHeight] (byte);

BDCTCr[BlockWidth, BlockHeight] (byte) – массивы значений компонент Y, Cb, Cr, восстановленные из битового куба до обратного DCT;

BY[BlockWidth, BlockHeight] (byte);

BCb[BlockWidth, BlockHeight] (byte);

BCr[BlockWidth, BlockHeight] (byte) – массивы значений компонент Y, Cb, Cr точек декодированного блока после обратного DCT;

BR[BlockWidth, BlockHeight] (sbyte);

BG[BlockWidth, BlockHeight] (sbyte);

BB[BlockWidth, BlockHeight] (sbyte) – массивы значений компонент R, G, B после преобразования из YCbCr, т.е. восстановленные компоненты точек блока изображения.

Данный набор полей предложен с учетом обработки блока, которая предусматривает возможность хранения промежуточных результатов; для «чистой» обработки можно обойтись тремя двумерными матрицами для представления исходного изображения, тремя матрицами для представления битового куба и тремя матрицами для представления восстановленного изображения.

Соответствующие методы данного класса реализуют алгоритмы обработки блоков:

void ToYCbCr() – преобразование компонент блока из RGB в YCbCr;

void DCT() – прямое DCT, применяемое для яркостной и цветоразностных компонент исходного изображения;

void FormBitCube() – формирование битового куба;

void RestoreBitCube() – «распаковка» битового куба;

void BDCT– обратное DCT, позволяющее получить яркостную и цветоразностную компонент точек восстановленного изображения;

void ToRGB - преобразование блока из YCbCr в RGB.

Передачу информации предлагается реализовать, используя методы класса Transfer. Этот класс будет иметь два поля:

TransferByte – байт передачи;

BufferName – имя файла, для хранения передаваемой последовательности.

Следует отметить, что хранение байта передачи предусмотрено и в классе Transfer и в классе BlockOfImage. Это позволяет реализовать вариант неравномерной компрессии изображения.

Передачу информации предлагается имитировать путем записи байтов сжатого изображения в файл, имя которого храниться в переменной BufferName.

Представление трансформанты в трехмерном пространстве позволяет организовать послойную передачу данных. Чтобы определить слои, которые необходимо передать, в работе [4] предлагается задать для каждой $k-$й трансформанты вектор передачи V_k, который соответствует двоичному представлению чисел от 0 до $p-1$:

$$V_k = \left\langle v_{k7}, v_{k6}, v_{k5}, v_{k4}, v_{k3}, v_{k2}, v_{k1}, v_{k0} \right\rangle .$$

Количество возможных вариантов передачи слоев битового куба P трансформанты будет равно 256 (2 в степени 8). Каждому варианту передаваемых слоев будет соответствовать изображение, соответствующее исходному, но различающиеся степенью искажения относительно исходного изображения.

Таким образом, после того, как вектор передачи задан, можно сразу из $k-$й трансформанты формировать передаваемую последовательность S_{kp} при заданном векторе передачи p:

$$S_{kp} = v_{k7} \,\&\, \ldots \,\&\, v_{km} \,\&\, \ldots \,\&\, v_{k0} \,\&\, \left(y_{ijm} \,\middle|\, v_{km} = 1 \right).$$

Каждый элемент представленной последовательности занимает 1 бит. В начало последовательности размещаются биты вектора передачи V_k, совокупность которых определяет номера слоев битового куба, которые будут переданы. Если m-й бит вектора передачи равен 1 (элемент v_{km}), то передаются все 64 элемента y_{ijm} слоя m. Если соответствующий бит вектора передачи равен 0, то значения бит данного слоя битового куба не передаются. В трансфер, кроме байта передачи следует передать старшие элементы «нулевых» элементов каждого блока. Алгоритм формирования последовательности для передачи $k-$й трансформанты при заданном векторе передачи p представлен в [4].

На приемной стороне выполняется операция сборки битового куба из последовательности S_{kp}. Суть данной операции заключается в том, что вначале извлекается вектор передачи (его образуют первые 8 бит принятой последовательности), кото-

рый затем анализируется следующим образом: если бит вектора передачи равен 1, то значениями следующих 64 бит заполняются ячейки соответствующего слоя битового куба; если элемент равен 0, то в куб добавляется слой из 0 элементов.

Алгоритм сборки битового куба на приемной стороне из полученной последовательности [4].

Соответственно для реализации указанных шагов следует предусмотреть соответствующие методы:

void Encode()

void SaveOnHDD(TransferByte)

void ReadFromHDD (TransferByte)

void Decode()

При реализации процедур компрессии и декомпрессии также предполагается вызов соответствующих методов класса BlockOfImage.

Модель процессов задачи представляет собой линейную структуру, которая предполагает последовательный вызов процедур соответствующей обработки блока.

Полученные описания моделей проектируемой программной компоненты вместе с алгоритмами формирования и распаковки передаваемой последовательности, представленными в [4], позволяют разработать соответствующие UML-диаграммы для ее непосредственной разработки.

Выводы

Представленные результаты проектирования позволяют утверждать, что разработка новых методов управления битовой скоростью определяет необходимость решения вопросов, связанных с дальнейшей их практической реализацией. Это достаточно сложная задача, требующая глубокой проработки.

Представленная логическая модель программного компонента, полученная на основе технологии объектного проектирования, позволяет достаточно полно представить необходимые элементы компрессии кадров видеопотока, а также позволяет реализовать обработку с использованием различных подходов.

Дальнейшим развитием результатов работы является кодирование и отладка соответствующего программного модуля и проведение исследований оперативности и качества компрессии изображений различных типов.

Список литературы

1. Беляев Е.А. Управление скоростью и ошибкой кодирования в системе сжатия и передачи видеоинформации с ограничениями на память передающего и принимающего устройств / Е.А. Беляев, А.М. Тюрликов // Компьютерная оптика. – 2007. – Т. 31, № 2. – С. 69-76.

2. Двухглавов Д.Э. Анализ подходов к управлению скоростью передачи видеопотока / Д.Э. Двухглавов, А.Г. Оксиюк, В.В. Твердохлеб // Сучасна спеціальна техніка: науково-практичний журнал. – 2014. – №2. – С. 29-35.

3. Двухглавов Д.Э. Методы контроля битовой скорости при компрессии видеоданных / Д.Э. Двухглавов, Н.А. Харченко // Системи обробки інформації. – Х.: ХУ ПС, 2014. – Вип. 9 (125). – С. 140-145.

4. Баранник В.В. Метод динамического управления битовой скоростью видеопотока с использованием трехмерного представления трансформант / В.В. Баранник, Д.Э. Двухглавов, В.В. Твердохлеб // Автоматизированные системы управления и приборы автоматики. – Х.: ХНУРЕ, 2014. – Вып.167. – С. 37-43.

5. Буч Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон. – СПб.: Питер, 2004. – 432 с.

6. JPEG [Электронный ресурс] – Режим доступа: http://www.pcs-ip.eu/index.php/main/edu/1.

Поступила в редколлегию 25.03.2016

Рецензент: д-р техн. наук, проф. В.В. Баранник, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.

ПРОЕКТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ КОМПРЕСІЇ НА ОСНОВІ БІТОВОГО ПРЕДСТАВЛЕННЯ БЛОКІВ ВІДЕОКАДРІВ

Д.Е. Двухглавов, М.М. Ніколаєв, В.В. Твердохліб

Представлені результати є етапом досліджень, які спрямовані на зменшення бітової швидкості обробки кадрів відеопотоку. Виконується проектування програмного компонента, який дозволить формувати бітові куби з початкового представлення блоку відеокадру, а потім бітову послідовність для передачі блоку, на основі обраного байту передачі, а також виконувати зворотне перетворення. Описано найбільш значущі для розробки даного компонента UML-діаграми, визначені відповідною специфікацією в парадигмі об'єктного підходу до проектування програмного забезпечення.

Ключові слова: *компресія відеокадрів, бітове представлення блоків відеокадру, об'єктне проектування програмного забезпечення, UML-діаграми.*

DESIGNING OF SOFTWARE IMPLEMENTATION OF COMPRESSION BASED ON THE BIT REPRESENTATION OF VIDEO FRAME UNITS

D.E. Dvukhglavov, N.N. Nikolaev, V.V. Tverdokhleb

The presented results are the research stage to reduce the bit rate of the video stream frame processing. Performs the designing of software component, that will form the bit cubes from the original presentation of the video frame unit, and then the bit sequence for the transmission unit, based on the selected byte of transfering, and to perform the reverse conversion. Described the UML-diagrams, that most important for the development of this component, that determined the appropriate specification in the paradigm of the objective approach to the software design.

Keywords: *compression of video frames, bit representation of video frame, object designing of software, UML-diagrams.*