УДК 621.396

Применение сеточного представления для сжатия графической информации

М.К. Чобану, О.С. Кургускин

Национальный исследовательский университет «МЭИ», cmk2@orc.ru, KurguskinOS@gmail.com

***Аннотация* — В статье описывается метод сжатия изображений и видео с помощью сеточного представления, основанного на триангуляции Делоне. Также описывается усовершенствование метода сжатия с помощью сеточного представления, описанного в предыдущих работах, проводится оценка слабых мест алгоритма. Кроме того в данной статье предложен новый подход к восстановлению цвета внутри треугольника.**

***Ключевые слова* — сеточное представление, триангуляция Делоне, сжатие графической информации.**

I. Введение

В настоящее время с созданием новых видеоинформационных систем и развитием сетей связи и вещания реализация новых эффективных методов сжатия и кодирования видеоинформации является важным направлением в современном мире. Существующие методы сжатия с потерями видеосигналов основаны на блочных технологиях кодирования [2], которые имеют существенный недостаток – блочный эффект. На то время, когда были предложены эти методы, блочная природа данных алгоритмов была весьма удобна в вычислительном отношении. Последние разработки в области сжатия видео вносили изменения, в основном, в размер блока, разрешение, а также в новые режимы предсказания для внутрикадрового кодирования. На сегодняшний день вычислительные мощности позволяют нам рассматривать новые более трудоемкие подходы к сжатию.

Известно, что реальное движение в кадре не является ни поступательным, ни блочно-постоянным. Модель, наиболее близкая к реальному движению, должна рассматривать изображения и видео как наборы объектов. Представление графической информации с помощью сеток, в свою очередь, дает более гибкий и удобный подход к оценке и компенсации движения в видео, чем существующие блочные методы. На рис. 1 схематично показано перемещение блоков при компенсации движения в кадре с помощью существующих блочных методов. На рис. 2 проиллюстрирована возможность улучшения решения задачи оценки и компенсации движения в видео с помощью сеток по сравнению с блочными методами. Можно заметить, что сеточный метод подчеркивает и форму объекта [5], и плавность его перемещения. Хотя также верно, что построение сеточного представления будет иметь более высокую вычислительную сложность, чем блочное. Для оценки возможности использования сеточных методов для сжатия графической информации были разработаны алгоритмы построения сеточного представления изображений, оценки и компенсации движения в видео и проведены оценки степени сжатия, качества и вычислительной сложности алгоритмов [1].

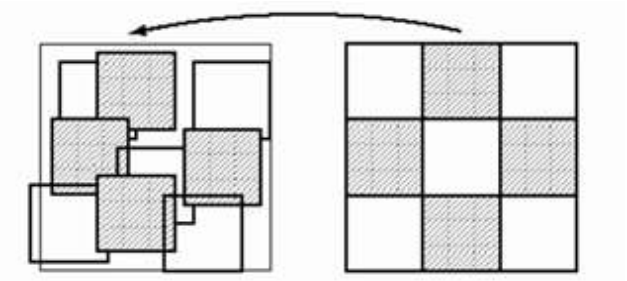

Рис. 1. Пример оценки движения с помощью блочного метода

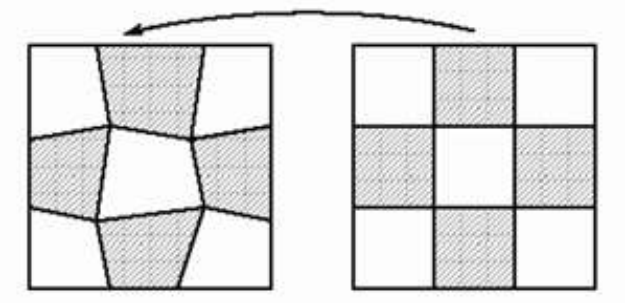

Рис. 2. Пример оценки движения с помощью сеточного метода

Разрабатываемый алгоритм сжатия графической информации основан на том, что изображение можно представить сеткой из треугольников – триангуляцией Делоне [3], вершинами которых выбраны наиболее важные точки изображения, после чего изображение можно будет восстановить, опираясь только на топологию сетки и значения цвета вершин. Тем самым можно получить сокращение объема информации, необходимого для хранения изображения, за счет представления его не набором цветов пикселей, а набором координат вершин сетки и цветов этих вершин. Важными точками на изображении считаются границы объектов, но т.к. все точки границы включать в список вершин не имеет смысла из-за их избыточно-

сти, то выделять нужно только некоторые особенные точки.

Для видеосигналов идея применения сеточных методов состоит в том, что после того, как кадр представлен треугольной сеткой, изменяя координаты узлов можно менять положение и форму треугольников сетки. Если выделить объекты на сеточном представлении и находить их смещение, можно будет получить оценку движения сетки. Для компенсации движения достаточно будет каждой точке объекта придать необходимое смещение и восстановить изображение.

В прошлой работе [1] был разработан метод сжатия изображений с помощью сеточного представления графической информации. Однако его реализация была недостаточно гибкая и быстрая для развития данного направления. Было принято решение изменить алгоритм и язык программирования, чтобы уйти от закрытых участков кода программы и перейти от стандартных средств к собственным, оставив общую концепцию метода сжатия. Для этого были произведены исследования «узких мест» алгоритма сжатия изображения, найдены пути их модернизации и проведены улучшения. Графики полученных изменений приведены ниже.

II. Анализ алгоритма

Основными проблемами разработанного алгоритма являлись время работы и восстановление цвета внутри треугольников. Существующую реализацию было трудно модернизировать, поэтому было решено изменить алгоритм, выявив те его части, которые нуждаются в изменении. Для выявления “узких мест” были поставлены таймеры на основные части алгоритма (сжатие изображения: применение фильтра Кенни, рекурсивный алгоритм разрежения матрицы границ, восстановление изображения: построение триангуляции, восстановление изображения). Части касающиеся оценки и компенсации движения было решено не замерять в связи с тем, что они нуждаются еще в логической переработке.

В результате были установлены усредненные процентные соотношения времен работы частей алгоритма:

1) Применение фильтров (2%) – небольшое размытие изображения (для снижения шума), повышения контрастности (для более четких границ), обесцвечивание (подготовка для фильтра Кенни), применение фильтра Кенни.

2) Разрежение матрицы границ (57%) – наиболее трудоемкая часть сжатия, ее сложность объясняется количеством рекурсивных вызовов.

3) Построение триангуляции (21%) – трудоемкость этого алгоритма напрямую зависит от количества точек. Поэтому необходимо найти алгоритм, трудоемкость которого была бы $O(N \cdot log(N))$ в худшем случае, в то время как трудоемкость использованного ранее алгоритма в худшем случае составляет $O(N^2)$.

Восстановление цветов треугольников (20%) – зависит от количества треугольников напрямую, однако порой встречаются случаи обработки пустых треугольников, что говорит о необходимости усовершенствования алгоритма разряжения матриц.

Главным направлением ускорения работы стал поиск решения задачи разрежения матрицы границ без использования рекурсивного алгоритма. Для этого было решено заменить рекурсивные вызовы последовательным перебором квадратов изображения устанавливаемой нечетной величины, размер которой влияет на степень сжатия, а также и на качество восстановления изображения. В полученном квадрате, если его размер более чем 3x3 пиксела, просматриваются границы. Если на сторонах выбранного «окна» лежат граничные точки, выбирается ближайшая к центру стороны точка и признается значимой, а также признается значимой ближайшая к центру «окна» точка, если такая есть, иначе выбирается центральная точка. Если отсутствуют точки на границах окна, происходит поиск граничных точек внутри «окна». Если их нет, то квадрат признается пустым, иначе ближайшая к центру квадрата признается значимой и вершины такого квадрата признаются значимыми точками.

Однако, учитывая обработку достаточно больших изображений (1920x1080 пикселей) квадраты разбиения обычно слишком малы и из-за этого будет слишком много значимых точек, а, следовательно, и степень сжатия будет незначительна. Так как на изображениях с большим разрешением достаточно часто встречаются обширные пустые области, которые не имеют внутри границ, то для них достаточно создать несколько точек, ограничивающих эту область.

Такой подход позволяет снизить затраты по времени, при этом создав адаптивную разреженную матрицу вершин наиболее близкую к геометрическому образу изображения.

На графике, представленном на рис. 3, показана зависимость времени работы алгоритма от плотности пикселей границ на битовой матрице границ. Видно, что предложенный алгоритм работает намного быстрее рекурсивного.

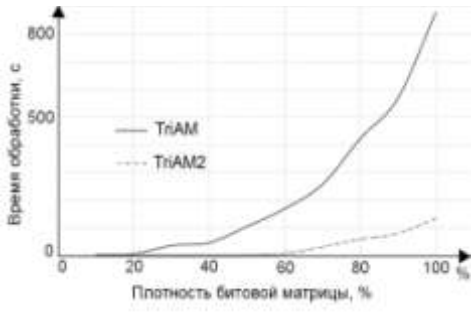

Рис. 3. Графики работы алгоритмов (рекурсивного и последовательного просмотра областей)

Вторым «узким местом» было построение триангуляции. Ранее оно обеспечивалось встроенной функцией MATLAB, для построения триангуляции исполь-

зуется алгоритм Quickhull [6, 7], который основан на инкрементальном алгоритме построения триангуляции с небольшими улучшениями (построение первоначального треугольника как выпуклой оболочки, использования дерева для хранения триангуляций).

S-hull – это новый $O(N \cdot log(n))$ алгоритм построения двухмерной триангуляции Делоне [8]. Главным компонентом алгоритма является радиально распространяющаяся развертка-остова (последовательно создается из радиально отсортированного набора 2D точек, давая непересекающуюся триангуляцию).

В эмпирических тестах алгоритм работает примерно в два раза быстрее, чем Q-hull для 2D триангуляции Делоне на случайно сгенерированных точечных множествах, это можно увидеть в таблице 1. В данной работе представлен алгоритм, который использует радиально распространяющуюся развертку-оболочку по радиально отсортированному множеству точек в сочетании с переворотом треугольников на последнем шаге.

Таблица 1

Время построения триангуляции от количества точек

Алгоритм	100 точек	1К точек	10К точек	100К точек	1М точек
Q-hull	0.0012 с	0,0115 с	0,109 с	1,96 с	24,3 с
S-hull	0.0007 с	0.0056 с	0.076 с	1.04 с	14,5 с

Задача восстановления изображения сводится к задаче интерполяции параметров на треугольнике. Пусть на экране задан треугольник с вершинами (i_1, j_1), (i_2, j_2), (i_3, j_3) и создана полутоновая палитра, из которой выбраны цвета c_1, c_2, c_3 в этих вершинах. На рис. 4 изображен рассматриваемый треугольник. Для определения цвета в произвольной точке (i, j), лежащей внутри треугольника, воспользуемся так называемыми барицентрическими координатами (α, β, γ) этой точки.

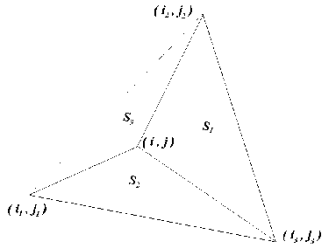

Рис. 4. Произвольный треугольник ABC

Геометрический смысл этих координат заключается в том, что они равны отношению площадей треугольников [10], изображенных на рис. 4.

Итак, алгоритм раскраски треугольника состоит из следующих шагов:

1) Вычислить:

$$D = (i_2 - i_1)(j_3 - j_1) - (i_3 - i_1)(j_2 - j_1) \ . \quad (1)$$

2) Вычислить коэффициенты:

$$A_1 = \frac{(i_3 - i_2)}{D};\ B_1 = \frac{(j_2 - j_3)}{D};\ C_1 = \frac{(i_2 j_3 - i_3 j_2)}{D} \ , \quad (2.1)$$

$$A_2 = \frac{(i_1 - i_3)}{D};\ B_2 = \frac{(j_3 - j_1)}{D};\ C_2 = \frac{(i_3 j_1 - i_1 j_3)}{D} \ . \quad (2.2)$$

3) Вычислить барицентрические координаты (α, β, γ) точки:

$$\alpha = B_1 \cdot i + A_1 \cdot j + C_1, \quad (3.1)$$

$$\beta = B_2 \cdot i + A_2 \cdot j + C_2, \quad (3.2)$$

$$\gamma = 1 - \alpha - \beta, \quad (3.3)$$

$$c = \alpha \cdot c_1 + \beta \cdot c_2 + \gamma \cdot c_3 \ , \quad (3.4)$$

где *c* – цвет искомой точки с координатами *(i, j)*.

Для определения принадлежности точки треугольнику используется алгоритм трассировки лучей [11].

III. Оценка алгоритма

Для оценки эффективности разработанного ранее метода исследовалась зависимость оценки количества бит на пиксель от качества восстановленного изображения (PSNR и PSNRHVS). Количество бит на пиксель (Bit Per Pixel, bpp) вычисляется как:

$$bpp = N_{bit}/(N_x \cdot N_y), \quad (3)$$

где N_{bit} – число бит, требуемое для хранения сжатого сообщения, N_x, N_y - размеры изображения в пикселях.

Распространённым объективным критерием оценки качества является СКО (среднеквадратическое отклонение, mean square error, MSE):

$$MSE = \frac{1}{N_x \cdot N_y} \sum_{i=0}^{N_x - 1} \sum_{j=0}^{N_y - 1} |I(i,j) - K(i,j)|^2, \quad (4)$$

где I и K – изображения размера высотой N_x и шириной N_y. Отношение пикового сигнала к шуму (или Peak signal-to-noise ratio, PSNR) определяется как:

$$PSNR = 20 \log_{10} \left(\frac{MAX_I}{MSE}\right), \quad (5)$$

где MAX_i — максимальное значение сигнала. В [9] предложен новый объективный критерий $PSNR_{HVS}$ на основе $PSNR$ путем введения таблицы корректирующих факторов.

Новое значение СКО определяется как:

$$MSE_{HVS} = \frac{\sum_{i=1}^{I-7} \sum_{j=1}^{J-7} \sum_{N_x}^{8} \sum_{N_y}^{8} \left((X_{ij} - X_{ij}{}^{e}) \cdot T_c\right)^2}{(I-7) \cdot (J-7) \cdot 64}. \quad (6)$$

При этом значение $PSNR_{HVS}$ вычисляется как:

$$PSNR_{HVS} = 10 \cdot \lg \left(\frac{255^2}{MSE_{HVS}}\right). \quad (7)$$

Оценки проводились на стандартном тестовом изображении Lena (256x256px).

Для варьирования bpp изменялся размер окна поиска значимых точек матрицы границ.

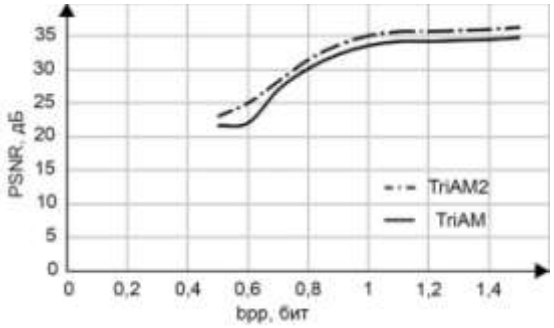

Рис. 5. Сравнение PSNR старого и модернизированного алгоритма

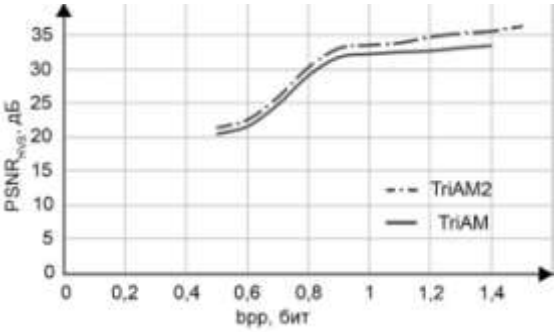

Рис. 6. Сравнение $PSNR_{HVS}$ старого и модернизированного алгоритма

Как видно из графиков, представленных на рис. 7 и рис. 8, усовершенствованный алгоритм имеет преимущество по сравнению с прошлой реализацией алгоритма. Однако алгоритм восстановления изображения стоит проработать для увеличения качества восстановленного изображения при низкой оценке *bpp*.

Из графика на рис. 8 видно, что усовершенствовав алгоритм удалось увеличить быстродействие. В основном, за счет замены рекурсивной функции последовательной.

Было произведено сравнение с одним из сеточных методов [4]. Для этого изображение в виде шахматной доски было сжато разными алгоритмами, в том числе и JPEG 2000. Результаты сжатия продемонстрированы на рис. 7.

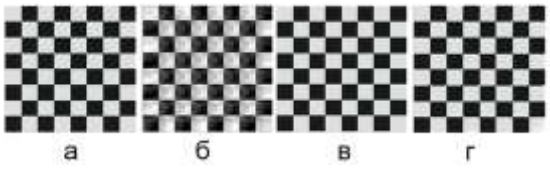

Рис. 7. Сравнение алгоритмов сжатия: а) оригинал, б) JPEG2000(PSNR=13дБ), в) AT2(PSNR=45дБ), г) TriAM(PSNR=25дБ)

Характеристики изображений при этом 256x256 в градациях серого, bpp ~ 0.23. Все оценки были получены на вычислительной машине с характеристиками: процессор – Intel i5 1.80 GHz; видеокарта: nVidia GeForce 740M; ОЗУ: 8G.

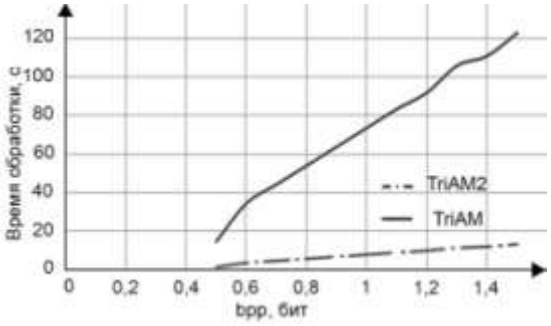

Рис. 8. Сравнение быстродействия алгоритмов

IV. Заключение

В результате исследований были найдены пути ускорения представления графической информации с помощью адаптивных сеток и был предложен усовершенствованный метод восстановления изображения.

Дальнейшие исследования будут направлены на создание гибридного метода оценки и компенсации видео. В гибридном методе оценка движения будет осуществляться с помощью распознавания изменения изображения в окрестности значимых точек, а компенсация движения будет происходить за счет применения изменений к узлам сетки и восстановления перестроенных треугольников.

Литература

[1] Чобану М.К., Кургускин О.С. Применение сеточного метода для сжатия графической информации. Основной алгоритм // Вестник компьютерных и информационных технологий. М.: Издательский дом "Спектр", 2013. Вып. 4. С. 13-18.

[2] Дворкович В.П. Чобану М.К. Проблемы и перспективы развития систем кодирования динамических изображений // MediaVision. 2011. Вып. 3. С. 58-60.

[3] Скворцов А.В. Триангуляция Делоне и ее применение. Томск: Изд-во Том. ун-та, 2002.

[4] URL:http://www.math.uni-hamburg.de/home/iske/papers/a4a6.pdf (дата обращения: 03.02.2014).

[5] URL: http://ldemaret.bplaced.net/homepage_ICB/di5.pdf (дата обращения: 03.02.2014).

[6] URL: http://www.mathworks.com/help/matlab/ref/delaunaytriangulationclass.html (дата обращения: 03.02.2014).

[7] URL:http://www.qhull.org/ (дата обращения: 03.02.2014).

[8] URL:http://www.s-hull.org/ (дата обращения: 03.02.2014).

[9] URL:www.soel.ru/cms/f/?/351062.pdf (дата обращения: 03.02.2014).

[10] URL: www.ru.convdocs.org/docs/index-9712.html (дата обращения: 03.02.2014).

[11]URL: www.algolist.manual.ru/maths/geom/belong/poly2d.php (дата обращения: 03.02.2014).