СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ СЖАТИЯ КАРТ ГЛУБИН БЕЗ ПОТЕРЬ

Бойцов Антон Владимирович, Гудков Владимир Юльевич, Цыбулевский Андрей Геннадьевич
Факультет Компьютерных технологий, управления и радиоэлектроники
Южно-Уральский Государственный Университет, Челябинск, Российская Федерация
antonboytsov@gmail.com, diana@sonda.ru, andrew.tsibulevsky@3divi.com

АННОТАЦИЯ

В статье представлен анализ методов сжатия без потерь статических изображений, представляющих собой карты глубин. Приведено описание основных методов сжатия данных без потерь и осуществлена модификация методов в соответствии с особенностями карт глубин. Собрана статистика по выполнению алгоритмов на тестовой выборке в различных разрешениях, определены алгоритмы, показывающие наилучшие характеристики как по коэффициенту, так и по времени сжатия.

Ключевые слова: *карта глубины, сжатие без потерь, алгоритм Хаффмана, арифметическое сжатие, алгоритм Лемпеля-Зива-Велча, алгоритм RLE.*

1. ВВЕДЕНИЕ

На протяжении своего развития и роста научно-технического прогресса, теория информации встречается с потребностью эффективно обрабатывать как привычные типы данных, так и информацию новой природы, например, снимки спутников и цифровых микроскопов. В настоящее время как в потребительской среде, так и в сфере исследований набирают популярность сенсоры глубины, позволяющие фиксировать трёхмерное состояние окружения. Информация о глубине объектов сцены может быть использована в различных целях как сама по себе, так и в сочетании с RGB-изображением обычной камеры. Доступ к информации о сцене предоставляется сенсором через генерацию последовательности карт глубин (рис. 1) — изображений, яркость каждого пикселя в которых соответствует расстоянию данной точки реального пространства до камеры сенсора глубины.

К областям применения карт глубин относятся выделение фона и переднего плана сцены, определение областей активности, трекинг пользователей, их поз и т. д. Также современные технологии позволяют использование сенсоров глубины в сочетании с технологиями 3DTV [8, 9] и виртуальной реальности, что позволяет пользователю, надевшему маску, видеть положение своего тела и взаимодействовать с объектами виртуального мира. Подобная технология подразумевает быструю передачу по проводным или беспроводным сетям карты глубины на маску либо смартфон пользователя, что, в свою очередь, касается задачи сжатия карты глубины.

Рисунок 1: Карта глубины

2. ПОСТАНОВКА ЗАДАЧИ

Задача кодирования изображения подразумевает выбор метода кодирования. Среди алгоритмов кодирования можно выделить семейства методов сжатия без потерь и с потерями. Сжатие без потерь производится в том случае, когда восстановленная информация должна полностью соответствовать исходной. Сжатие с потерями используется, если малые потери информации (например, о заднем плане сцены) не критичны для восприятия изображения после распаковки. Оба метода кодирования активно применяются по отношению к разным типам изображений [1].

Также если говорить о кодировании последовательности изображений, то алгоритмы можно разделить на межкадровые и внутрикадровые. Межкадровое сжатие использует факт, что изображение на двух или нескольких соседних кадрах практически не меняется. В таком случае

задача кодирования кадра может сводиться к кодированию разности последовательности кадров. Если метод сжатия избегает этой особенности, ему необходимо кодировать кадр целиком, что подразумевает внутрикадровое сжатие. У обоих методов есть свои преимущества — при межкадровом сжатии зачастую достигается максимальная эффективность кодирования видео, однако оно невозможно, если, например, при трансляции по сетям могут происходить потери кадров.

В данной статье рассмотрены наиболее известные внутрикадровые алгоритмы сжатия информации без потерь, для каждого из методов измерена эффективность сжатия как по коэффициенту, так и по времени компрессии и декомпрессии. Также в работе показано, что использование основных особенностей карт глубин в реализации алгоритмов значительно повышает эффективность сжатия.

3. ОСОБЕННОСТИ КАРТЫ ГЛУБИНЫ

Изображение, представляющее собой карту глубины, имеет существенные отличия от обычных 24-битных RGB-изображений. Изображение, полученное с сенсора Microsoft Kinect является 1-канальным 16-битным изображением, каждый пиксель которого может принимать значения от 0 до 65535 мм, что соответствует расстоянию до точки в реальном мире. Однако стоит заметить, и не все значения из этого диапазона используются, поскольку сеснор считывает глубину с динамической дискретизацией — чем дальше объект от камеры, тем выше дискретность [6]. Поэтому каждой глубине на карте можно сопоставить её порядковый номер — «слой». Карта слоёв, полученная таким образом, в дальнейшем поможет оптимизировать стандартные алгоритмы сжатия.

Кроме того, у сенсора есть ограничения на максимальную возможную глубину объекта — она составляет 10000 мм. Вследствие этих причин, количество возможных значений глубины не превышает 1000, а на практике конкретный кадр и вовсе может содержать менее 200 различных глубин.

Также карта глубины имеет так называемые «тени» — области с глубиной 0 рядом с границами объекта. Они появляются за счёт того, что камера и инфракрасный излучатель находятся на некотором расстоянии друг от друга. Значение глубины 0 появляется также в тех пикселях, расстояние до которых выше, чем максимальное (10000 мм) [6].

Ещё одной важной особенностью карты глубины является представление объектов на карте в виде гладких поверхностей [7]. Это означает, что соседние точки на изображении имеют схожую глубину. Исключение составляют ситуации, когда эти точки принадлежат разным объектам (происходит разрыв на границе), либо объект находится под большим углом к камере.

4. ТЕСТОВЫЕ ИЗОБРАЖЕНИЯ

Тестовая выборка, по которой оценены результаты выполнения алгоритмов, представляет собой набор из 85 карт глубин, взятых из последовательностей, самостоятельно записанных с помощью 3D-сенсора Microsoft Kinect® и представленных в виде изображений оттенков серого. Глубина цвета на них составляет 16 бит, изображения хранятся в файлах формата TIFF (без сжатия). На картах глубины присутствуют люди и различные объекты на расстоянии до 10 метров от камеры сенсора (рис. 2). Оригинальные изображения имеют разрешение 640×480, однако при тестировании каждая карта глубины приводится к размерам 320×240, 160×120, 80×60 и 40×30 пикселей.

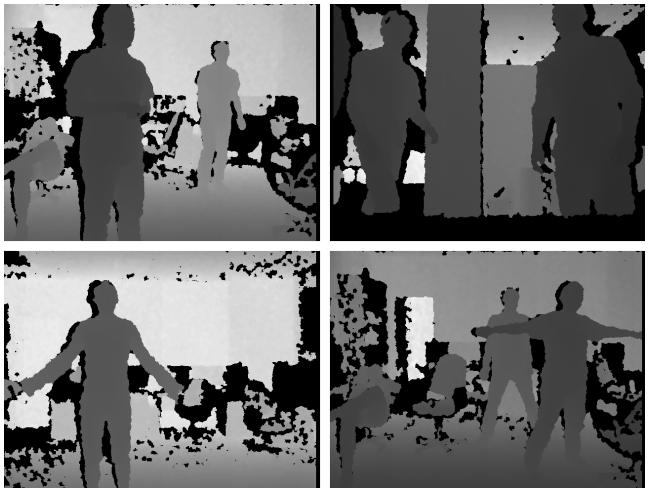


Рисунок 2: Примеры тестовых изображений

5. АЛГОРИТМ ХАФФМАНА

Один из самых первых эффективных алгоритмов кодирования информации был предложен Д.А. Хаффманом в 1952 г. Суть алгоритма заключается в подсчёте частоты символов в сообщении и присваивании часто встречающимся символам кодов меньшей длины, а редко встречающимся — большей. Здесь и в дальнейшем символами применительно к карте глубины будем считать значения глубины, присутствующие на изображении [2].

Первым этапом сжатия является подсчёт частоты значений глубин на изображении и построение по этой статистике дерева Хаффмана. Каждый лист дерева Хаффмана соответствует некоторому значению глубины, а проходящий по разным вершинам путь от корня до листа — коду переменной длины. Дерево Хаффмана, как и любое бинарное дерево, может быть представлено в обычном массиве. Кроме того, во время построения дерева необходимо сохранять соответствие между значениями глубин и кодами, чтобы в дальнейшем выполнить кодирование — эту информацию можно хранить в структуре map (или словарь) из стандартной библиотеки C++. После построения дерева за один проход карте глубины можно произвести кодировку всех символов — вместо глубины глубины в буфер вывода необходимо записать набор бит, являющийся её кодом. Стоит отметить, что перед кодировкой карты необходимо в качестве вспомогательной информации передать в буфер вывода размеры изображения и наборы «глубина — длина кода — битовый код» для всех возможных значений глубины (рис. 3).

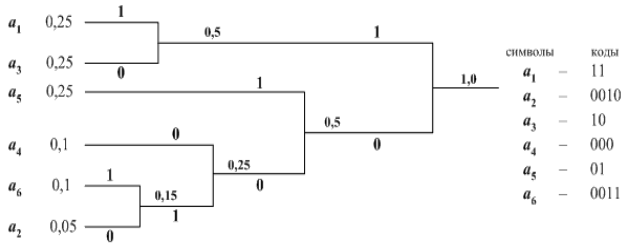


Рисунок 3: Пример дерева Хаффмана

Перед декодированием из переданной вспомогательной информации необходимо сформировать словарь, в котором, наоборот, каждой кодовой последовательности будет соответствовать значение глубины. Процесс декодирования будет проходить следующим образом: каждый считанный бит присоединяется к текущей последовательности, после чего производится проверка наличия этой последовательности в словаре. Как только последовательность найдена, в ячейку изображения помещается соответствующая глубина. При этом стоит заметить — алгоритм Хаффмана гарантирует, что ни один код не является префиксом другого, поэтому первая присутствующая в словаре последовательность бит является искомым кодом.

Кроме того, в качестве более эффективной модификации сжатия по Хаффману возможно кодирование не самих значений глубины, а их разностей на карте слоёв. Кодирование разностей на карте слоёв дополнительно уменьшает размерность алфавита, потому что соседние слои на дальнем и ближнем расстоянии будут иметь одну и ту же разность, хотя у глубин на этих расстояниях разности будут различаться. Сжатие разностей слоёв показывает лучшие результаты как среди изображений малого разрешения, так и высокого. В дальнейшем эта техника будет применяться во всех алгоритмах сжатия.

6. АРИФМЕТИЧЕСКОЕ КОДИРОВАНИЕ

Идея арифметического сжатия заключается в том, чтобы представить информацию в виде вещественного числа в промежутке 0 до 1. Первый этап кодирования заключается в том, чтобы посчитать долю каждого символа в тексте и разделить отрезок [0; 1) на подотрезки в соответствии с получившимися значениями (рис). При этом каждому символу будет присвоен свой подотрезок. Дальнейшее кодирование происходит следующим образом — считывается символ и рассматривается подотрезок, соответствующий данному символу. Этот подотрезок становится текущим и делится на те же доли, что исходный отрезок [0; 1). Далее считывается следующий символ последовательности, и новый подотрезок выбирается внутри текущего. Таким образом, к концу последовательности символов остаётся отрезок, любое число из которого (например, середину) можно интерпретировать как код для всей последовательности.

Декодирование происходит аналогично — отрезок [0; 1) делится на доли, на каждом шагу выбирается символ, подотрезок которого содержит в себе число-код, этот отрезок нормируется и операция повторяется до тех пор, пока не декодируется необходимое число символов (рис. 4).

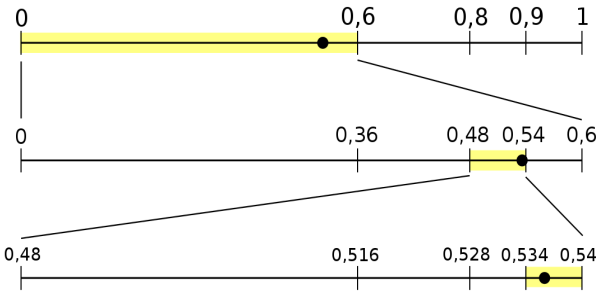


Рисунок 4: Иллюстрация арифметического сжатия

При кодировании большого последовательности символов (что относится и к изображениям) получившееся вещественное число с высокой вероятностью не войдёт ни в один существующий стандартный тип, поэтому на практике биты записываются в выходной буфер сразу после вычисления, а для поддержания текущего отрезка выполняется нормировка [5].

7. АЛГОРИТМ ЛЕМПЕЛЯ-ЗИВА-ВЕЛЧА

Алгоритм Лемпеля-Зива-Велча (LZW) является одним из самых эффективных алгоритмов сжатия без потерь в плане степени сжатия. Его главная идея в том, что код присваивается не каждому символу в отдельности, а целым последовательностям символов. В начале N кодов (от 0 до N - 1) присвоены только одиночным символам алфавита, которые присутствуют в передаваемом сообщении. При чтении сообщения накапливаются последовательности. Если на каком-то шаге последовательность не имеет соответствующего кода, то она добавляется в конец списка, в буфер вывода записывается код последовательности без последнего символа, а её последний элемент становится началом новой последовательности [3].

Аналогично происходит декодирование, для осуществления которого необходимо иметь лишь наборы кодов для одиночных символов, при этом коды последовательностей формируются по ходу декодирования [4].

8. АЛГОРИТМ RLE

Run-length encoding (RLE) — простой и в некоторых случаях достаточно эффективный алгоритм кодирования последовательностей. Его суть заключается в замене ряда повторяющихся символов специальным кодом, состоящим из самого символа и количества его повторений [4]. В случае сжатия карт глубин стандартная реализация RLE-кодирования заведома неэффективна, поскольку отрезки с одинаковым значением глубин присутствуют редко. Однако есть два значимых факта:

1. «нулевая глубина», обозначающая отсутствие глубины в пикселе, часто сгруппирована в некоторых местах изображения;
2. разность между соседними глубинами на участках карты, соответствующих поверхностям, изменяется монотонно.

Учитывая эти факты, можно выделить три режима кодирования:

1. кодирование глубины по номеру на карте слоёв;
2. кодирование нулевых глубин по RLE;
3. кодирование близких значений глубин через разности на карте слоёв.

Стоит отметить, что код для RLE будет состоять из единственного n-битного числа, которое будет задавать количество подряд идущих нулей — максимум 2^n. Если же количество нулей превышает эту величину, n-битовый код будет повторяться ещё раз, и так до тех пор, пока последовательность нулей не закончится. Для разных разрешений карт глубины максимально эффективное сжатие наблюдается при выборе n равным от 2 до 5.

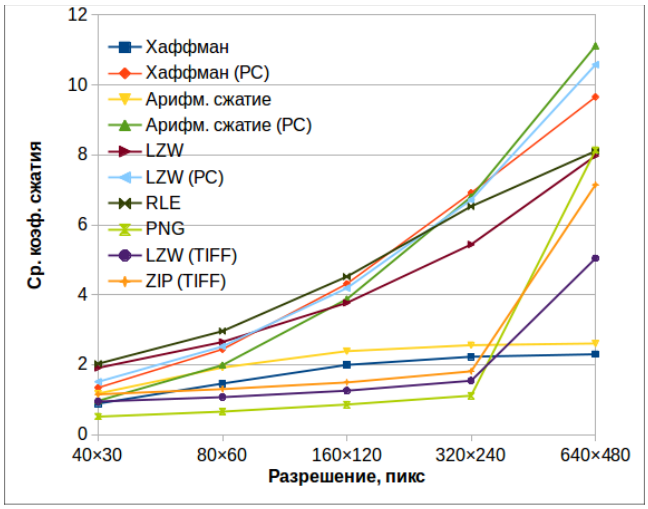
В свою очередь, код для разностей на карте слоёв представляет собой m-битовое число, кодирующее 2^m-1 отрицательных и неотрицательных значений разности (и одно значение, обозначающее смену режима кодирования). Для достижения максимальной эффективности сжатия в зависимости от размера карты m следует выбирать равным 2 или 3.

9. СТОРОННИЕ ПРИЛОЖЕНИЯ

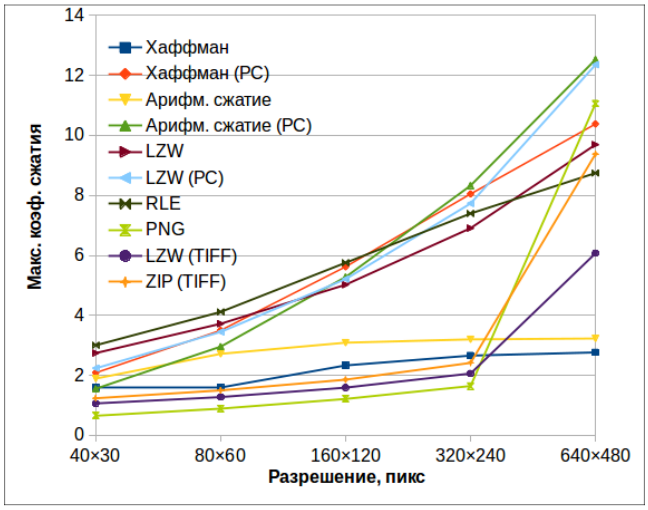
Для более объективной оценки качества сжатия карт глубины необходимо провести сравнение рассмотренных методов сжатия с алгоритмами, широко используемыми на практике в различных программных продуктах. В этих целях осуществлено сжатие тестовой выборки методами, поддерживаемыми широко распространённым пакетом для обработки графики ImageMagick. Программа ImageMagick позволяет конвертировать изображения между различными форматами (PNG, JPEG, TIFF, GIF и т.д.) в разных режимах. Для сравнения с алгоритмами, реализованными самостоятельно, было проведено преобразование без потерь в форматы PNG и TIFF (алгоритмами LZW и ZIP) 16-битного изображения оттенков серого.

10. ЗАКЛЮЧЕНИЕ

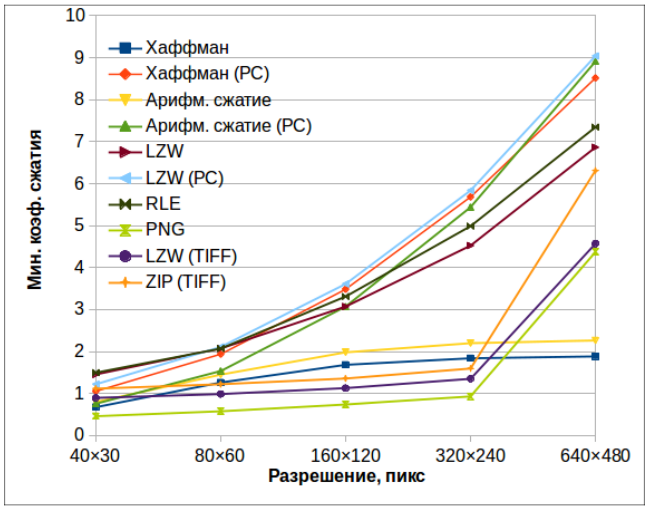
В данной статье были проанализированы различные методы сжатия карт глубин без потерь. Сравнительные результаты тестирования алгоритмов можно наблюдать на рисунках 5-7.



(а)

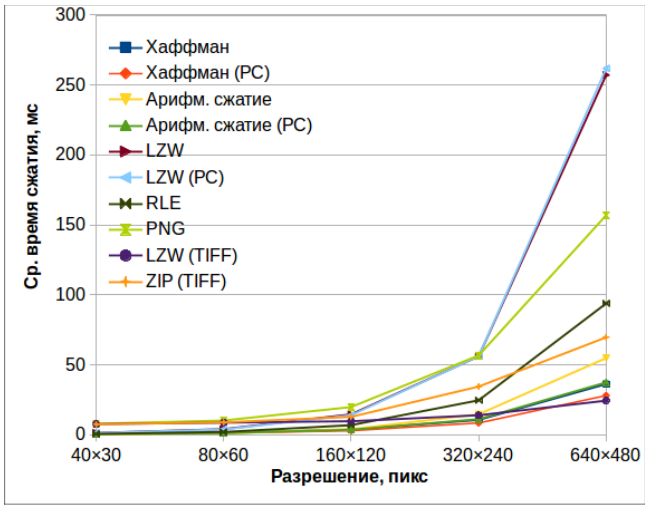


(б)

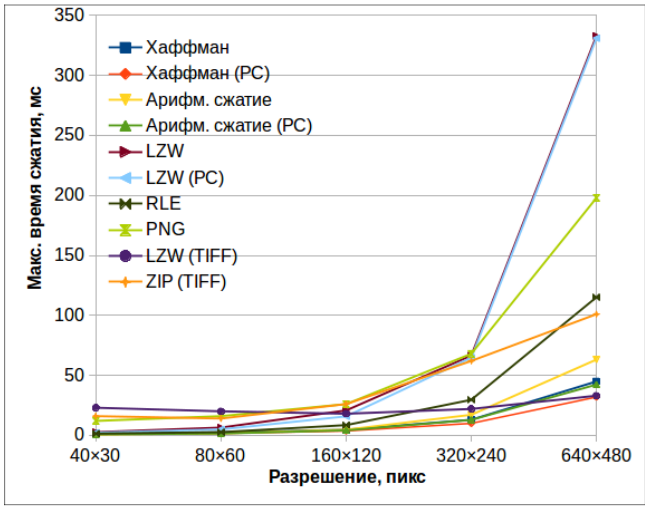


(в)

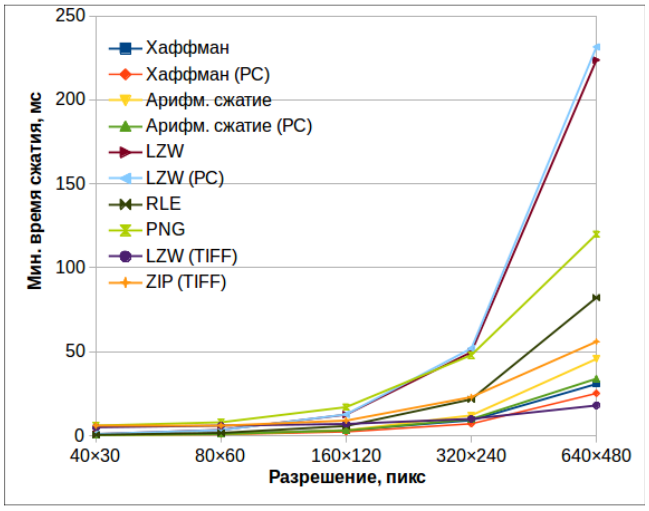
Рисунок 5: Средний (а), максимальный (б) и минимальный (в) коэффициент сжатия карт глубины. Сокращение «PC» соответсвует сжатию разности слоёв



(а)

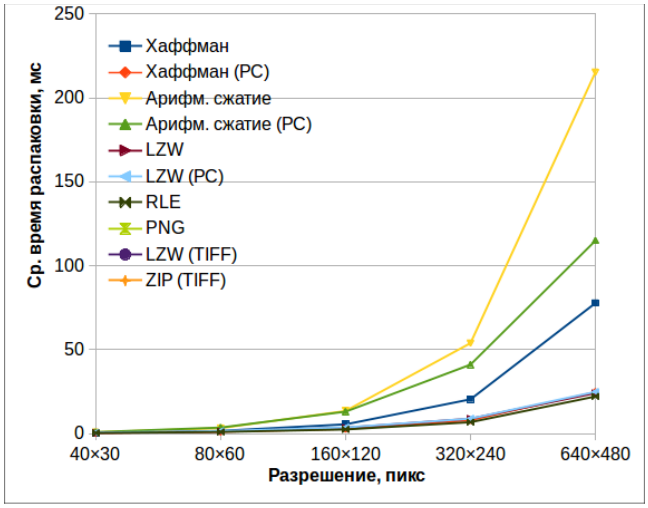


(б)

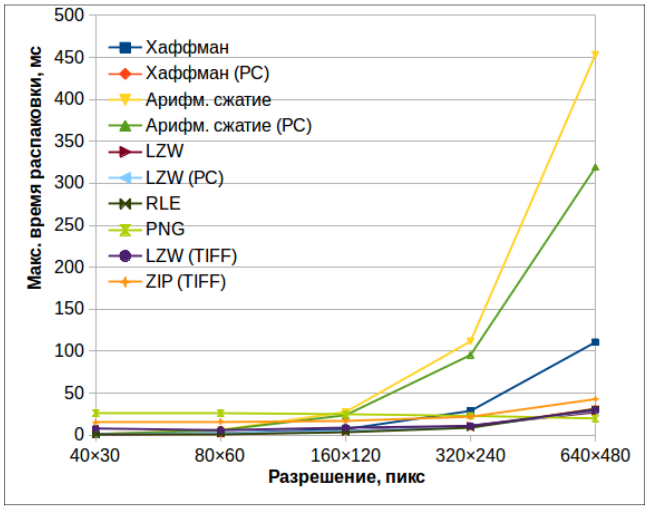


(в)

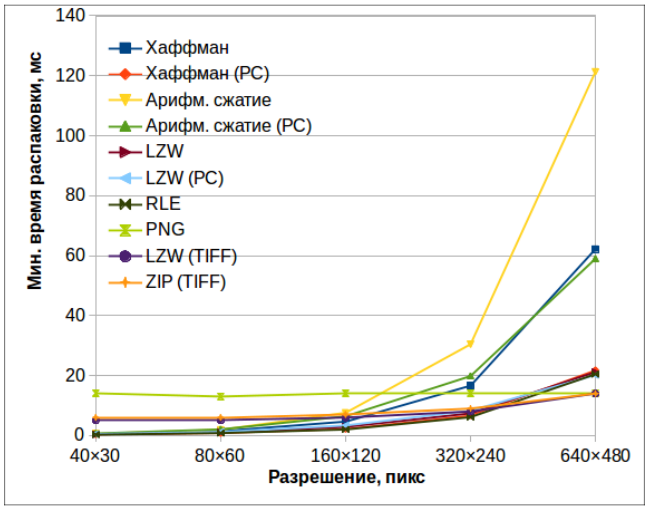
Рисунок 6: Среднее (а), максимальное (б) и минимальное (в) время сжатия карт глубины



(а)



(б)



(в)

Рисунок 7: Среднее (а), максимальное (б) и минимальное (в) время распаковки карт глубины

Наилучших показателей достигли модифицированные методы, учитывающие различные особенности карты глубины. Алгоритмы, использующиеся в пакете ImageMagick показывают достаточно высокий коэффициент сжатия лишь на картах более высокого разрешения. Все измерения проводились на процессоре Intel® Core™ i3-2350M, имеющем 4 ядра частотой 2.30 Ггц.

Наиболее эффективные методы сжатия для карт глубин разных разрешений можно наблюдать в таблице 1. На малых разрешениях (от 40×30 до 160×120) лучшие результаты как по степени сжатия, так и по времени, показывает алгоритм RLE. На разрешениии 320×240 алгоритмы Хаффмана, арифметического сжатия, LZW (все — на разностях слоёв), а также RLE показывают примерно одинаковый лучший результат. На максимальном разрешении лучший средний коэффициент сжатия показывает алгоритм арифметического сжатия разностей слоёв, но близкие показатели имеют также алгоритмы LZW и Хаффмана. Стоит отметить, что существует пространство для оптимизации (в том числе с использованием параллельных вычислений) по времени как компрессии, так и декомпрессии для самостоятельно реализованных алгоритмов.

Размер	Метод	Ср. коэф. сжатия	$t_{код}^{ср}$, мс	$t_{дек}^{ср}$, мс
40×30	RLE	2,025	0,702	0,461
80×60	RLE	2,959	2,081	1,054
160×120	LZW (PC)	4,517	6,936	2,573
320×240	Хаффман (PC)	6,914	8,509	7,981
320×240	Арифм. сжатие (PC)	6,799	11,237	41,183
320×240	LZW (PC)	6,714	56,22	9,046
320×240	RLE	6,526	24,692	6,898
640×480	Арифм. сжатие (PC)	11,117	37,542	115,015
640×480	LZW (PC)	10,579	261,894	24,948
640×480	Хаффман (PC)	9,656	28,093	24,948

Таблица 1: Наилучшие алгоритмы сжатия для карт глубины разных разрешений

11. ССЫЛКИ

[1] Кудряшов, Б.Д. *Теория информации: Учебник для вузов,* СПб.: Питер, 2009.

[2] Huffman, D. *A method for the construction of minimum-redundancy codes.* Proceedings of the IRE, 40(9):1098–1101, 1952.

[3] Mahoney, M.. *Data Compression Explained* – http://mattmahoney.net/dc/dce.html, дата обращения 04.04.16.

[4] Nelson, M., *The Data Compression Book*, M&T Books, Redwood City, CA, 1996

[5] Nelson M., *Data Compression with Arithmetic Encoding,* http://www.drdobbs.com/cpp/data-compression-with-arithmetic-encodin/240169251, дата обращения 29.04.16

[6] S. Mehrotra, Z. Zhang, Q. Cai, C. Zhang, and P.A. Chou. *Low-Complexity, Near-Lossless Coding of Depth Maps from Kinect-Like Depth Cameras.* IEEE MMSP 2011, 2011.

[7] J. Gautier, O.L. Meur, C. Guillemot. *Effcient depth map compression based on lossless edge coding and diffusion.* In: Picture Coding Symposium, Krakow, Poland 81–84, 2012.

[8] K. Samrouth, O. Deforges, Y. Liu, F. Pasteau, M. Khalil and W. Falou. *Efficient depth map compression exploiting correlation with texture data in multiresolution predictive image coders.* Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW), pp. 1-6 , 2013.

[9] S. Hoffmann, M. Mainberger, J. Weickert, M. Puhl, A. Kuijper, K. Bredies, T. Pock and H. Bischof. *Compression of depth maps with segment-based homogeneous diffusion.* Scale-Space and Variational Methods in Computer Vision, vol. 7893, pp. 319-330, 2013.