30. Ingaramo D., Errecalde M., Cagnina L., Rosso P. Particle Swarm Optimization for lustering short-text corpora. *Computational Intelligence and Bioengineering*. Eds F. Masulli, A. Micheli, A. Sperduti. IOS Press, 2009, pp. 3–19.
31. Azzag H., Monmarche N., Slimane M., Venturini G. AntTree: A new model for clustering with artificial ants. *Proc. of the 2003 Congress on Evolutionary Computation (CEC '03)*. IEEE Press, 2003, vol. 4, pp. 2642–2647.
32. Stein B., Meyer zu Eißen S. Document Categorization with MAJORCLUST. *Proc. of the 12th Workshop on Information Technology and Systems (WITS 02)*. Eds A. Basu, S. Dutta. Barcelona, Spain, Technical University of Barcelona, 2002, pp. 91-96.

Попова Светлана Владимировна – инженер, ст. преподаватель, Санкт-Петербургский государственный университет; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Санкт-Петербург, Россия, svp@list.ru

Данилова Вера Владимировна – аспирант, Автономный университет Барселоны, Барселона, Испания, maolve@gmail.com

Svetlana Popova – engineer, senior lecturer, Saint Petersburg State University; Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint Petersburg, Russia, svp@list.ru

Vera Danilova – postgraduate, Autonomous University of Barcelona, Barcelona, Spain, maolve@gmail.com

УДК 004.925.4

ИСПОЛЬЗОВАНИЕ КОНТЕЙНЕРА BC7 ДЛЯ ХРАНЕНИЯ ТЕКСТУР С ГЛУБИНОЙ ЦВЕТА 10 БИТ

И.В. Перминов[a], Т.Т. Палташев[a, b, c]

[a] Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Санкт-Петербург, Россия, i.am.perminov@gmail.com
[b] Северо-Западный политехнический университет, Фримонт, Калифорния, США, timpal@mail.npu.edu
[c] Advanced Micro Devices (AMD), Калифорния, США, timpal@mail.npu.edu

Изображения с высокой глубиной цвета обладают гораздо лучшими возможностями воспроизведения цветов и плавных градиентов, особенно при использовании устройств с широким цветовым охватом. В работе рассматривается проблема сжатия текстур с глубиной цвета 10 бит, применяемых в трехмерной компьютерной графике. Предложен метод хранения подобных текстур с использованием стандартного формата сжатия текстур BC7, рассмотрен тип блока BC7, с помощью которого можно закодировать цвета с точностью, превышающей 8 бит, и показаны необходимые изменения в аппаратуре декодера. Предложенный подход обладает обратной совместимостью с существующими декодерами. В ходе исследования была разработана программная реализация кодека на основе компрессора bc7_gpu. Сравнение исходного и предлагаемого кодека показало уменьшение ошибок сжатия для изображений с высокой глубиной цвета.

Ключевые слова: сжатие текстур, BC7, block compression, глубина цвета, Direct3D.

USAGE OF BC7 CONTAINER FOR STORING TEXTURES WITH 10-BIT COLOR DEPTH

I. Perminov[d], T. Paltashev[d, e, f]

[d] Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint Petersburg, Russia, i.am.perminov@gmail.com
[e] Northwestern Polytechnic University, Fremont, California, USA, timpal@mail.npu.edu
[f] Advanced Micro Devices (AMD), California, USA, timpal@mail.npu.edu

High color depth images can more accurately reproduce colors and smooth color transitions without banding artifacts, especially for wide color gamut devices. The paper deals with texture compression with 10-bit color depth applied in 3D graphics. A method for storing of such textures in standard BC7 blocks is proposed. The example of BC7 block applicable for storing of smooth texture information with an accuracy exceeding 8-bit is also given. The proposed approach has a backward compatibility with current hardware. Additional hardware cost to support 10-bit decoding is expected to be low. An overview of

developed software for compressor based on bc7_gpu is given. Comparison of PSNR and RMSE shows that the proposed compressor provides better quality for 10-bit textures; compressing errors become less for images with high color depth.
Keywords: texture compression, BC7, block compression, color depth, Direct3D.

Введение

Стандартным методом представления изображений в компьютерных системах является TrueColor, подразумевающий использование цветовой модели RGB с глубиной цвета 8 бит, поэтому интенсивности каждого цветового канала (красного, зеленого и синего) могут принимать значения от 0 до 255. Хотя приложения могут использовать для хранения и обработки изображений прочие форматы с повышенной точностью, вывод на экран, как правило, осуществляется в формате TrueColor. При этом стандартным цветовым пространством является sRGB [1]. Цветовое пространство определяет, какой реальный цвет соответствует каждому конкретному числовому значению цвета. И хотя пространство sRGB покрывает всего около 35% всех видимых человеком цветов [2], глубины цвета TrueColor не хватает для отображения всех цветов даже внутри sRGB [3]. Наибольшие проблемы возникают с отображением плавных градиентов, так как в некоторых случаях явно видны границы между соседними цветами. Данная проблема усугубляется при использовании более широких цветовых пространств, например AdobeRGB, где доступные в каждом цветовом канале 256 уровней еще сильнее отдаляются друг от друга.

Исходя из этого, в профессиональных сферах применения, таких как работа с медицинскими снимками, кинопроизводство, графический дизайн, обработка фотографий, зачастую используются мониторы с поддержкой глубины цвета 10 бит и более и соответствующие форматы представления цвета. Полноценная поддержка таких изображений при работе с трехмерной графикой также является важной задачей.

Текстуры в трехмерной компьютерной графике

Использование различных видов текстур является неотъемлемой частью современной компьютерной графики. Это позволяет существенно улучшить визуальную детализацию трехмерных объектов без усложнения геометрии. В простейшем случае текстуры представляют собой двухмерное изображение, накладываемое на трехмерный объект. Отдельные точки растра, составляющие текстуру, принято называть текселями (texel – texture element). Однако применение большого количества текстур крупных размеров предъявляет высокие требования к видеопамяти.

В связи с этим широкое распространение получили различные технологии сжатия текстур [4–8]. При этом текстура передается и хранится в памяти в сжатом виде, а распаковывается непосредственно на графическом процессоре, как правило, между кэшами L1 и L2. Такой подход позволяет не только уменьшить занимаемый текстурами объем памяти, но и сэкономить пропускную способность, которая в большинстве случаев является куда более дефицитным ресурсом. Таким образом, использование текстурного сжатия позволяет существенно повысить производительность системы визуализации. Кроме этого, использование сжатия положительно сказывается на энергопотреблении за счет уменьшения объема используемой памяти и снижения трафика на интерфейсе «графический процессор» ↔ «видеопамять».

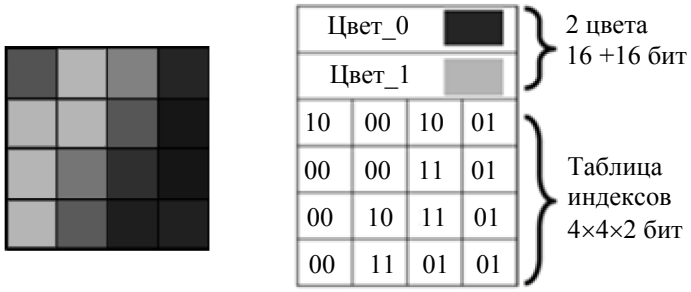
Универсальные алгоритмы сжатия без потерь (RLE, LZW, Deflate) и стандартные методы сжатия изображений плохо подходят для текстур, так как при отрисовке трехмерных объектов крайне важен эффективный доступ к произвольному участку текстуры. По этой причине большинство существующих форматов сжатия – это блочные кодеки с фиксированным уровнем сжатия. При этом изображение разбивается на множество блоков одинакового размера, которые сжимаются независимо друг от друга. Размер отдельного блока, как правило, составляет 4×4 текселя.

В большинстве случаев текстуры сжимаются в режиме «offline», так как алгоритмы, позволяющие снизить ошибки сжатия, обладают высокими вычислительными затратами. Но существуют и алгоритмы быстрого сжатия [9], однако сжатие динамически генерируемых текстур ограничивается применяемыми программными моделями работы с текстурами [10].

Современные видеоадаптеры и программные интерфейсы OpenGL [11] и Direct3D [12] уже достаточно давно обладают поддержкой текстур с большой глубиной цвета. Однако возможность сжатия таких текстур отсутствует. Исключение составляют текстуры, хранящие изображение с широким динамическим диапазоном, или HDR (High Dynamic Range). Отметим, что можно использовать существующие форматы для хранения изображений с глубиной цвета 10 бит.

Формат сжатия текстур BC7

Форматы сжатия текстур, используемые в Direct3D 10 и Direct3D 11, называются BlockCompression. Всего доступно семь различных форматов, начиная с BC1 и заканчивая BC7, а все семейство форматов часто обозначают как BCn. Во всех форматах сжатия BCn используется блок размером 4×4 текселя, и применяются схожие базовые принципы компрессии, которые проще всего продемонстрировать на примере блока BC1 (рис. 1).



а **б**

Рис. 1. Пример блока BC1: исходный блок 4×4 пикселя размером 512 бит (а); сжатый блок размером 64 бита (б)

В сжатом блоке (рис. 1, б) сохраняются 2 ключевых цвета в формате RGB565 (по 5 бит на красный и синий цветовые каналы, и 6 бит на зеленый канал) и таблица индексов, указывающая, в какой пропорции смешивать ключевые цвета при распаковке. Так как размер индекса составляет 2 бита, то внутри блока доступно только 4 различных цвета.

Формат BC7 [13, 14] появился в Direct3D 11. Блок BC7 занимает 128 бит и имеет гораздо более сложную структуру, чем BC1. Одним из основных отличий BC7 от BC1 является возможность сохранения до трех пар базовых цветов. Первые биты в блоке кодируют его тип. Всего существует восемь типов блоков, отличающихся набором и размером полей. В некоторых блоках, кроме цветовых каналов R, G и B, также сохраняется значение альфа-канала, что может, к примеру, использоваться для реализации эффекта полупрозрачности. Кроме этого, блоки отличаются размером таблицы индексов, количеством пар базовых цветов, точностью их представления и прочими полями. Размер индекса варьируется от 2 до 4 бит, а глубина цвета – от (4+1) бит до (7+1) бит. Подобная запись (7+1) используется потому, что дополнительный бит во всех цветовых каналах одной базовой точки обозначается как P и имеет одинаковое значение.

Распаковка сжатого блока осуществляется аппаратно. При этом, согласно требованиям Direct3D, результат работы аппаратного декодера BC7 должен побитово совпадать с результатом эталонного декодера. Упрощенно распаковка выглядит следующим образом: сначала осуществляется преобразование базовых цветов в формат с точностью представления 8 бит на канал (деквантование), после этого путем их смешивания с помощью линейной интерполяции формируется локальная палитра цветов, и индекс из таблицы используется для выборки из этой палитры.

Модификация этапов кодирования и декодирования

Идея предлагаемого подхода заключается в использовании 10-битного представления ключевых цветов и локальной палитры при распаковке, а также в модификации компрессора. При этом сам формат сжатого блока остается прежним.

Дело в том, что самые проблемные для 8-битной глубины места, а именно блоки с очень плавным изменением цвета, могут быть закодированы в BC7 с точностью, превышающей 8 бит. Для этого, к примеру, может использоваться тип блока BC7 Mode6 (рис. 2). В данном типе блока размер одного индекса составляет 4 бита (один бит в таблице индексов кодируется неявно). Фактически можно считать, что в данном блоке точность представления базовых цветов составляет 8 бит, а 4-битный индекс позволяет указать один из двух базовых цветов или любое из 14 промежуточных значений. Если значения базовых цветов достаточно близки, что как раз соответствует плавному градиенту, то промежуточные значения позволяют закодировать цвета, недоступные для глубины 8 бит. Если же декодер будет оперировать значениями с большей точностью, то подобные цвета смогут быть воспроизведены на выходе корректно. В связи с этим можно ввести новый режим работы декодера, в котором деквантование базовых цветов и последующие операции будут производиться с точностью 10 бит.

Аппаратная реализация дополнительного режима распаковки, выдающего на выходе значения с глубиной цвета 10 бит, не составит большого труда, так как логика работы декодера не изменится. В то же время сжатые текстуры остаются полностью совместимыми с обоими декодерами, так как формат сжатого блока остается прежним. Распаковка текстуры с глубиной 10 бит на «старом» декодере будет просто соответствовать понижению глубины цвета до 8 бит. И наоборот, текстуры с глубиной 8 бит при распаковке на предлагаемом декодере могут иметь лучшее качество по сравнению с обычной распаковкой и дальнейшим преобразованием до глубины цвета 10 бит за счет повышения точности представления промежуточных цветов в локальной палитре.

Естественно, для корректного сжатия текстур с глубиной цвета 10 бит необходима модификация компрессора, потому что существующие кодеры BC7 [15–17] принимают на вход изображения с точностью 8 бит.

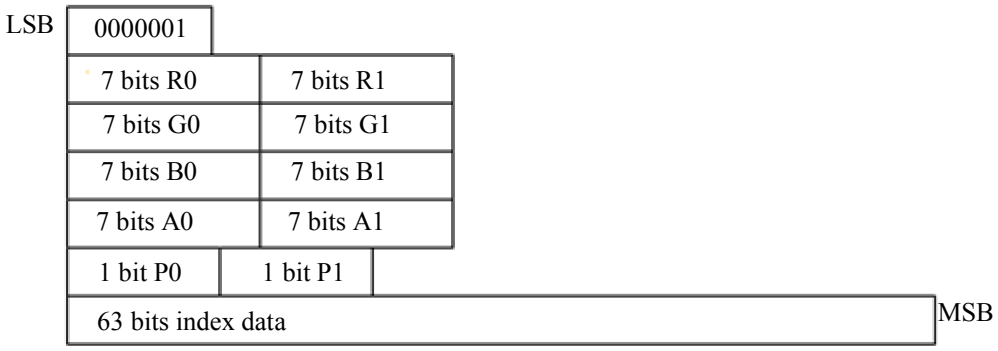


Рис. 2. Формат блока BC7 Mode6. LSB (Least Significant Bit) – младший значащий бит, MSB (Most Significant Bit) – старший значащий бит

Программная реализация

Для практической проверки состоятельности предложенного метода было произведено тестирование качества сжатия различных текстур с глубиной цвета 10 бит с использованием программного кодека. За основу был взят кодек bc7_gpu [15], производящий сжатие текстуры в формат BC7 силами графического процессора с использованием технологий CUDA [18] и OpenCL [19].

Произведенные модификации заключались в изменении типов данных для возможности хранения цвета с повышенной точностью и модификации функций квантования/деквантования. Логика работы компрессора не изменялась.

Для тестирования качества сжатия были выбраны текстуры с глубиной цвета 10 бит и наличием плавных градиентов. При сжатии текстур с помощью оригинального компрессора производилась предварительная конвертация исходного изображения в формат с представлением цвета TrueColor. После распаковки с использованием оригинального декодера производилось преобразование в формат с точностью 10 бит. Для этого значения в каждом цветовом канале пересчитывались согласно формуле (1), где c – интенсивность цветового канала. Данная формула соответствует сдвигу на два разряда влево и копированию двух старших бит в высвободившиеся младшие разряды. Это стандартная целочисленная аппроксимация для деквантования, используемая, в том числе и в аппаратных блоках графических процессоров:

$$c = (c << 2) \mid (c >> 6). \quad (1)$$

Значения метрик ошибок сжатия RMSE (Root Mean Square Error, среднеквадратическое отклонение) и PSNR (Peak Signal to Noise Ratio, соотношение сигнал/шум) для тестовых текстур приведены в таблице. Ошибки сжатия для текстуры № 1 также продемонстрированы на рис. 3.

	Оригинальный кодек		Предлагаемый кодек	
	RMSE	PSNR	RMSE	PSNR
Текстура № 1	175,2	51,4	144,7	53,1
Текстура № 2	178,3	51,3	150,4	52,8
Текстура № 3	170,1	51,7	99,3	56,4
Текстура № 4	249,9	48,3	222,3	49,4

Таблица. Качество сжатия текстур с глубиной цвета 10 бит

Результаты тестирования показывают, что предлагаемая модификация ощутимо увеличивает качество сжатия текстур, имеющих плавные градиенты с глубиной цвета 10 бит.

Заключение

Таким образом, предложенный метод позволяет использовать уже существующие форматы сжатия для хранения текстур с высокой глубиной цвета. При аппаратной модификации декодера это позволит получить при использовании таких текстур все преимущества сжатия, включая повышение производительности, снижение энергопотребления и требований к подсистеме памяти. При этом текстуры остаются обратно совместимыми с существующими декодерами, которые смогут корректно распаковать блок с использованием глубины цвета 8 бит. Также стоит отметить, что последующая модификация логики компрессора для работы с глубиной цвета 10 бит позволит еще сильнее снизить ошибки сжатия.

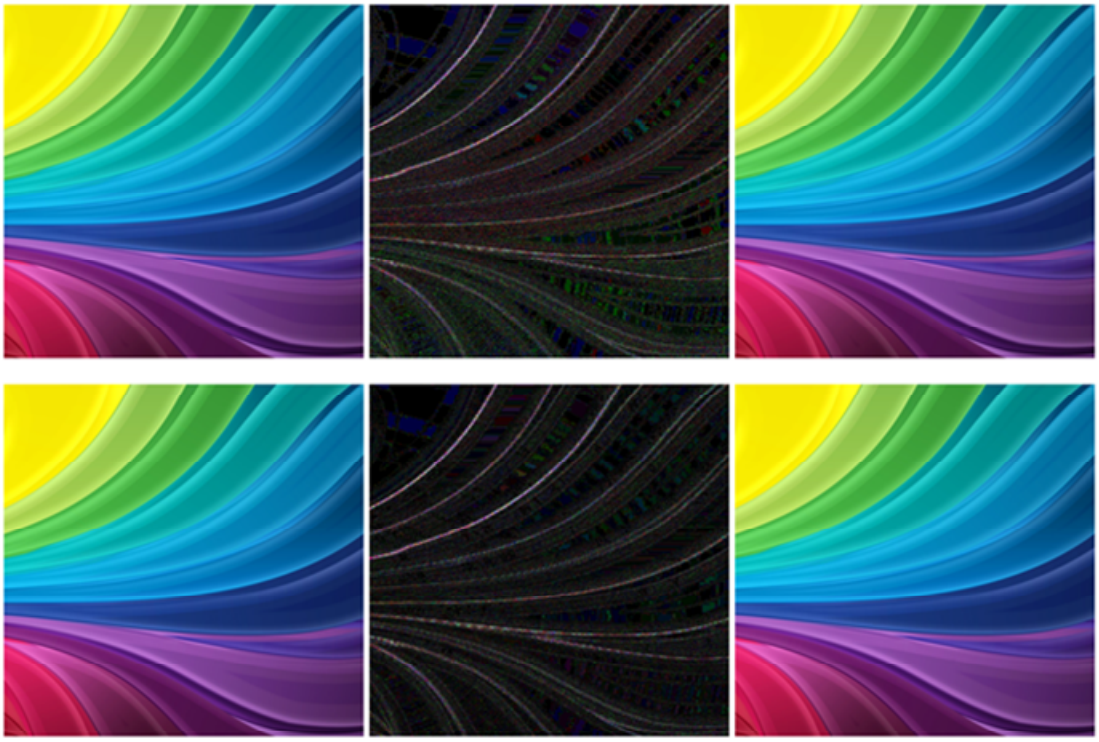


Рис. 3. Пример сжатия тестовой текстуры № 1, размер текстуры 512×512 пикселей

References

1. Stokes M., Anderson M., Chandrasekar S., Motta R. *A Standard Default Color Space for the Internet – sRGB*. Available at: http://www.w3.org/Graphics/Color/sRGB.html (accessed 23.11.2013).
2. *SRGB VS. ADOBE RGB* 1998. Available at: http://www.cambridgeincolour.com/tutorials/sRGB-AdobeRGB1998.htm (accessed 23.11.2013).
3. *Is 8 bits enough? Of course not.* Available at: http://19lights.com/wp/2011/09/30/is-8-bits-enough-of-course-not/ (ac-cessed 23.11.2013).
4. Hong Z., Iourcha K.I., Nayak K.S. *System and method for fixed-rate block-based image compression with inferred pixel values.* Patent US 5956431 A. Filing date: 02.10.97. Publication date: 21.09.99.
5. Fenney S. Texture compression using low-frequency signal modulation. *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware.* Eurographics Association, 2003, pp. 84–91.
6. Strom J., Akenine-Moller T. iPACKMAN: High-Quality, Low-Complexity Texture Compression for Mobile Phones. *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware.* Eurographics Association, 2005, pp. 63–70.
7. Strom J., Pettersson M. ETC2: Texture Compression using Invalid Combinations. *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware.* Eurographics Association, 2007, pp. 49–54.
8. Nystad J., Lassen A., Pomianowski A., Ellis S., Olson T. Adaptive scalable texture compression. *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics.* Eurographics Association, 2012, pp. 105–114.
9. van Waveren J.M.P. *Real-time DXTcompression.* Id Software, Inc., 2006, 42 p.
10. Perminov I.V. Povyshenie effektivnosti obrabotki dinamicheski szhimaemykh tekstur [Improvement of dynamic texture compression]. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics,* 2013, no. 6 (88), pp. 164–165.
11. *OpenGL 4.4 Core Specification.* Available at: http://www.opengl.org/registry/doc/glspec44.core.pdf (accessed 23.11.2013).
12. *Direct3D 11 Graphics.* Available at: http://msdn.microsoft.com/enus/library/windows/desktop/ff476080.aspx (accessed 23.11.2013).

13. *BC7 Format Mode Reference.* Available at: http://msdn.microsoft.com/enus/library/windows/desktop/hh308954.aspx (accessed 23.11.2013).
14. *OpenGL A.R.B. ARB texture compression bptc.* 2010. Available at: http://www.opengl.org/registry/specs/ARB/texture_compression_bptc.txt (accessed 23.11.2013).
15. *Free high-quality BC7 GPU texture compressor.* Available at: http://gpuscience.com/software/free-high-quality-bc7-gpu-texture-compressor/ (accessed 23.11.2013).
16. *DirectXTex texture processing library.* Available at: http://directxtex.codeplex.com/ (accessed 23.11.2013).
17. Krajcevski P., Lake A., Manocha D. FasTC: Accelerated Fixed-Rate Texture Encoding. *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D'13)*. NY, ACM, 2013, pp. 137–144.
18. *CUDA Parallel Computing Platform.* Available at: http://www.nvidia.com/object/cuda_home_new.html (accessed 03.12.2013).
19. *OpenCL – The open standart for parallel programming of heterogeneous systems.* Available at: http://www.khronos.org/opencl/ (accessed 03.12.2013).

Перминов Илья Валентинович – аспирант, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Санкт-Петербург, Россия, i.am.perminov@gmail.com

Палташев Тимур Турсунович – доктор технических наук, профессор, зав. кафедрой; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Санкт-Петербург, Россия; профессор, Северо-Западный политехнический университет, Фримонт, Калифорния, США; начальник отдела моделирования и исследования графических архитектур, Advanced Micro Devices (AMD), timpal@mail.npu.edu

Ilya Perminov – postgraduate, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint Petersburg, Russia, i.am.perminov@gmail.com

Timour Paltashev – D.Sc., Professor, Department head, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint Petersburg, Russia; Professor, Northwestern Polytechnic University, Fremont, California, USA; Senior Manager, Graphics Architecture Research and Modeling, Advanced Micro Devices (AMD), timpal@mail.npu.edu

УДК 621.397.3

ПРЕДШЕСТВУЮЩАЯ И ПОСЛЕДУЮЩАЯ ФИЛЬТРАЦИЯ ШУМОВ В АЛГОРИТМАХ ВОССТАНОВЛЕНИЯ ИЗОБРАЖЕНИЙ[1]

В.С. Сизиков[a, b], Р.А. Экземпляров[b]

[a] Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Россия, Санкт-Петербург, sizikov2000@mail.ru

[b] Санкт-Петербургский государственный политехнический университет, Санкт-Петербург, Россия, rexe@yandex.ru

Рассмотрено зашумление смазанных или (и) дефокусированных изображений. Определяется последовательность фильтрации шумов на таких изображениях – до устранения смазывания/дефокусирования или после него. Введены понятия предшествующей и последующей фильтрации шумов. Устранение смазывания/дефокусирования ряда изображений выполнено методами параметрической фильтрации Винера и регуляризации Тихонова, а фильтрация шумов – методами медианной фильтрации Тьюки и адаптивной фильтрации Винера. На репрезентативных выборках проведена экспериментальная проверка, получены количественные оценки погрешностей восстановления изображений при различных типах шумов и очередностей их фильтрации. Показано, что методы параметрической фильтрации Винера и регуляризации Тихонова достаточно эффективно устраняют смазывание/дефокусирование, но недостаточно фильтруют шумы. Эффективность фильтрации шумов повышается при добавлении таких методов, как медианный фильтр Тьюки, адаптивный фильтр Винера. При этом для импульсного шума важен порядок (очередность) его фильтрации (до или после устранения смазывания/дефокусирования в зависимости от помехо-сигнальной ситуации), а для гауссова шума порядок несущественен. Показано, что импульсный шум лучше фильтруется медианной, ранговой, адаптивной медианной фильтрацией, а гауссовый шум – адаптивной винеровской фильтрацией, среднеарифметическим фильтром. Дается объяснение этим эффектам.

Ключевые слова: зашумленное смазанное или дефокусированное изображение, предшествующая и последующая фильтрация шума, устранение смазывания и дефокусирования.

[1] Работа выполнена при поддержке РФФИ (грант № 13-08-00442).