УДК 681.325

ЦВЕТОВОЕ ПРЕОБРАЗОВАНИЕ ДЛЯ СЖАТИЯ КОМПЬЮТЕРНЫХ И СИНТЕТИЧЕСКИХ ИЗОБРАЖЕНИЙ БЕЗ ПОТЕРЬ

В. О. Минченков,
аспирант
А. В. Сергеев,
ведущий программист
А. М. Тюрликов,
канд. техн. наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения

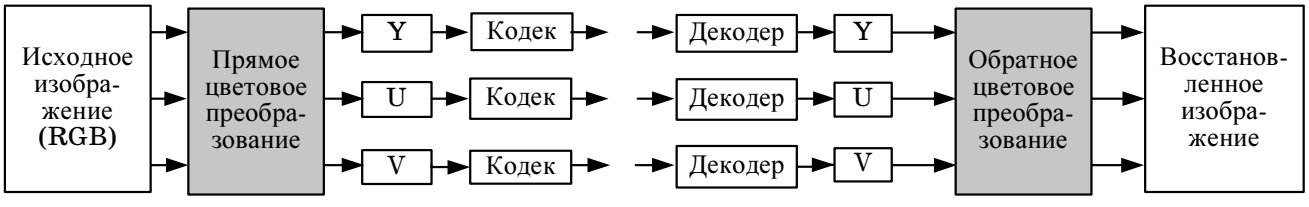
Представлено расширенное цветовое преобразование (Extended YUV), позволяющее существенно увеличить эффективность сжатия компьютерных и синтетических изображений без потерь по сравнению со стандартными преобразованиями. Предлагаемый алгоритм обладает малым уровнем сложности. Основные идеи подхода продемонстрированы с использованием YUV(YCrCb)-преобразования и кодеков JPEG-LS и H.264/AVC.

Введение

Элементы современных систем сжатия изображений (H.264/AVC [1], JPEG 2000 [2]) ориентированы, большей частью, на работу с фотореалистичными изображениями. При сжатии компьютерной графики их эффективность сильно падает. Цветовое преобразование является важным элементом большинства кодеков (JPEG, H.264/AVC, JPEG 2000 и т. д.) и предназначено для уменьшения зависимости между компонентами изображения. Стандартные преобразования, например широко известное YUV, крайне плохо справляются с подобной задачей при обработке компьютерных изображений с цветным текстом, иконками, многочисленными цветовыми переходами и т. д. Авторами предлагается расширение стандартного цветового преобразования YUV (E-YUV), позволяющее решить указанную проблему и значительно увеличить степень сжатия изображений подобного рода.

Цветовое преобразование в системах сжатия изображений

Общая схема системы сжатия неподвижных изображений представлена на рис. 1. Как видно на схеме, исходное изображение в самом начале попадает на блок прямого *цветового преобразования*. Его основное предназначение — уменьшить зависимость между исходными компонентами R, G и B, преобразовывая их в новые, менее зависимые между собой компоненты (в данном примере — Y, U и V). Для того чтобы пояснить назначение цветового преобразования, будем предполагать, что на множестве исходных изображений RGB задано какое-то распределение вероятности и соответственно может быть вычислена энтропия такого вероятностного ансамбля. Аналогичное допущение сделаем и для множества YUV. В идеальном случае цветовое преобразование обладает двумя свойствами. Во-первых, является взаимообратным и не вносит каких-либо искажений, т. е. вы-



■ ***Рис. 1.*** *Структура кодека неподвижных изображений*

полняется равенство $H(RGB) = H(YUV)$. Во-вторых, для новых компонент выполняется равенство $p(yuv) = p(y)p(u)p(v)$, т. е. полученные компоненты абсолютно независимы. Отсюда вытекает свойство $H(YUV) = H(Y) + H(U) + H(V)$, которое объясняет возможность кодирования компонент независимо друг от друга.

Для большинства существующих цветовых преобразований можно сказать, что $H(YUV) \le H(Y) + H(U) + H(V)$, так как зависимость между компонентами остается, хотя и меньше чем в исходном RGB-представлении. Следовательно, достижимо более эффективное сжатие изображения в целом. Точно проверить данное неравенство не представляется возможным, так как не известны распределения на RGB и на YUV, и, следовательно, судить о наличии зависимостей между компонентами можно только на качественном уровне.

Целью данного раздела было предложить такое цветовое преобразование, которое делало бы компоненты «менее зависимыми», и при этом не пришлось бы вносить значительных изменений в существующие схемы сжатия изображений при его применении.

Эффект уменьшения зависимостей между компонентами достигается за счет введения добавочной, «четвертой» компоненты, в которой будет храниться дополнительная информация о фоне изображения. То есть попытаемся выполнить следующее равенство (сумма энтропий компонент, полученных после преобразования, должна быть приблизительно равна взаимной энтропии исходных компонент):

$$H(RGB) = H(YUV) \approx$$
$$\approx H(Y) + H(U') + H(V') + H(Y''U''V'') \le$$
$$\le H(Y) + H(U) + H(V).$$

На практике в реальных системах сжатия (MPEG4, H.264/AVC, JPEG) в качестве блока предварительной обработки используется YUV (YCbCr)-преобразование. Поэтому оно взято в данной работе для объяснения основных принципов предлагаемого подхода.

Цветовые преобразования

Цветовое пространство представляет собой модель представления цвета, основанную на использовании цветовых координат. Цветовое пространство строится таким образом, чтобы любой цвет был представим точкой, имеющей определенные координаты, причем так, чтобы одному набору координат (и точке пространства) соответствовал один цвет. Количество координат задает размерность пространства.

Наиболее распространенное пространство — RGB [3, 4]. Это трехмерное цветовое пространство, где каждый цвет описан набором из трех координат — каждая из них отвечает компоненте цвета в разложении на красный (R, Red), зеленый (G, Green) и синий (B, Blue) цвета.

Существует масса цветовых пространств различной размерности — от одномерных, которые могут описать исключительно монохромное изображение (например, PAL D/K , PAL B/G), до шести- и десятимерных, таких, например, как пространство CMYKLcLm (Cyan, Magenta, Yellow, blacK, lightCyan, lightMagenta). Выбор цветового пространства определяется областью применения:

1) печать и полиграфия (CMYK [5], CMYKLcLm, RGB, PMS, NCS);

2) обработка и редактирование (XYZ, HSV [6], LAB);

3) хранение и передача (YUV, YCbCr, YFbFr, YPbPr [7], YCoCg-R [8], RCT).

Пространства высокой размерности чаще всего используют для печати и полиграфии, а пространства малой размерности — для хранения и передачи изображений.

Цветовым преобразованием принято называть уравнения перехода из пространства RGB (считается базовым) в какое-либо иное цветовое пространство.

Существуют различные уравнения переходов из одного пространства в другое с определенной долей точности, так как пространства могут не иметь однозначных отображений друг в друга. Так, к примеру, пространство RGB можно отобразить в пространство YUV и обратно с некоторыми потерями, получив около 60 дБ отношения сигнал/шум в среднем. Например, значения компонент RGB = (0,3,3) после преобразования RGB → YUV → → RGB примут значения RGB = (1,2,4).

В данной работе нас будут интересовать только те преобразования, которые в основном используются в алгоритмах сжатия изображений (JPEG, MPEG, H.264/AVC), а именно RGB, YUV(YCbCr) и YCoCg.

Независимость компонент изображения

Чтобы пояснить необходимость использования пространств типа YUV в задачах хранения и передачи изображений, следует, в первую очередь, прокомментировать недостатки представления изображений в пространстве RGB.

Значения соседних пикселей в цветовом пространстве RGB обычно похожи и сильно зависимы. Другими словами, одни и те же детали изображения представлены во всех трех компонентах, т. е. все исходное изображение можно наблюдать на каждой компоненте в отдельности, как, например, на рис. 2 (см. с. 3 обложки). Если вычислить коэффициенты попарной взаимной корреляции компонент R, G и B в одном изображении, то они примут очень большие значения (табл. 1):

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \overline{A}\right)\left(B_{mn} - \overline{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \overline{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \overline{B}\right)^2\right)}}.$$

■ ***Таблица 1.*** *Значения попарной корреляции между компонентами*

Компоненты	Значение взаимной корреляции
RG	0,9921
RB	0,9637
GB	0,9859
Сумма RGB	**2,9417**
YCr	0,0902
YCb	0,0546
CrCb	0,8653
Сумма YCrCb	**1,0101**

В современных алгоритмах сжатия кодирование компонент происходит независимо (т. е. при сжатии не учитывается зависимость между сигналами). При этом наличие взаимной зависимости между компонентами (как в случае RGB) приводит к уменьшению возможной степени сжатия. При удалении взаимозависимости может значительно увеличиться степень сжатия. Следовательно, чем больше взаимная энтропия, от которой избавляются с помощью нового преобразования, тем значительнее выигрыш по степени сжатия новых компонент по сравнению со стандартными компонентами RGB.

Указанная выше избыточность информации в компонентах RGB отрицательно сказывается на эффективности сжатия изображений. Вследствие чего на начальном этапе алгоритмов сжатия переводят изображение из пространства RGB в другое пространство, где компоненты более независимы между собой.

Для этой цели в кодеках используются линейные цветовые пространства, такие как YUV (YCbCr), YCoCg и т. д., основным свойством которых является слабая зависимость между компонентами и маленькие значения корреляции между компонентами. Как видно на рис. 3 (см. с. 3 обложки), компоненты, полученные после преобразования YCbCr, являются менее информативными (гладкими), чем исходные R, G и B, а в табл. 1 можно увидеть, что и значения взаимной попарной корреляции для этих компонент значительно меньше.

YUV (YCbCr)-преобразование

YUV-преобразование [9] используется уже достаточно долго в различных системах, связанных со статическими и динамическими изображениями. С 1950 г. — в стандартах аналогового телевидения NTSC и PAL/SECAM. Компонента Y идеально подходила для передачи сигнала для телевизоров, способных отображать лишь черно-белое изображение. В цветном телевизоре дополнительные сигналы U и V позволяли восстанавливать всю картинку в цвете. С приходом цифровой обработки данных преобразование YUV нашло применение в алгоритмах сжатия изображений JPEG, MPEG4, H.264/AVC и в бурно развивающемся сегодня широкоформатном телевидении (HDTV). Но для цифровых систем более корректно использовать дискретный аналог YUV, YCbCr со значениями компонент, выравненными в диапазоне [0, 255]. В данной статье термины YUV и YCbCr синонимичны. Сегодня также широко применяются аналогичные YUV цветовые преобразования, такие как YFbFr и YCoCg. Эти преобразования имеют ту же смысловую нагрузку, но в отличие от YUV не имеют ошибок округления, так как работают с целочисленными коэффициентами (подробнее рассмотрим далее).

В пространстве YUV (YCbCr)-компоненты имеют следующие обозначения: Y — яркостная компонента (характеристика освещенности точки); U, V (Cr, Cb) — цветоразностные компоненты (chrominance).

Прямое преобразование из RGB в YCbCr:

$$Y = 0{,}299R + 0{,}587G + 0{,}114B;$$

$$Cb = -0{,}168736R - 0{,}331264G + 0{,}5B + 128;$$

$$Cr = 0{,}5R - 0{,}418688G - 0{,}081312B + 128.$$

Обратное преобразование из YCbCr в RGB:

$$R = Y + 1{,}402(Cr - 128);$$

$$G = Y - 0{,}7141(Cr - 128) - 0{,}34414(Cb - 128);$$

$$B = Y + 1{,}772(Cb - 128).$$

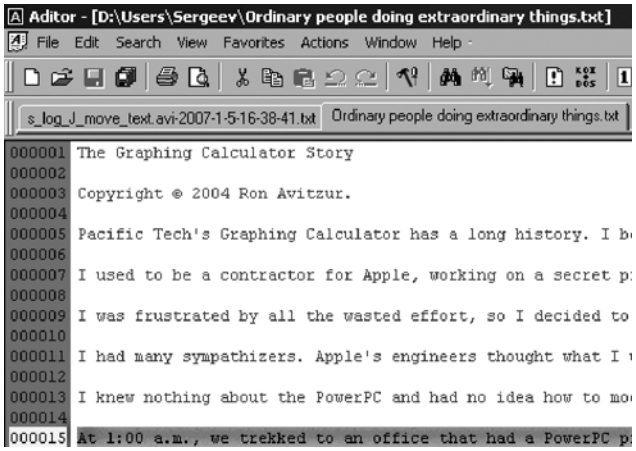
Основные проблемы существующих линейных цветовых преобразований

Преобразование YCrCb в среднем дает хорошие результаты при разложении фотореалистичных изображений на компоненты, т. е. получаемые на выходе компоненты на кодеках имеют высокий коэффициент сжатия. Но на синтетических и компьютерных изображениях, содержащих цветной текст, разноцветные иконки и т. п., коэффициент сжатия падает сильнее, чем можно было ожидать, так как в основном в таких изображениях имеется меньшее количество деталей, чем в фотоизображениях. В действительности если рассмотреть компоненты YCrCb от такого синтетического изображения, то можно наглядно наблюдать большую избыточность информации.

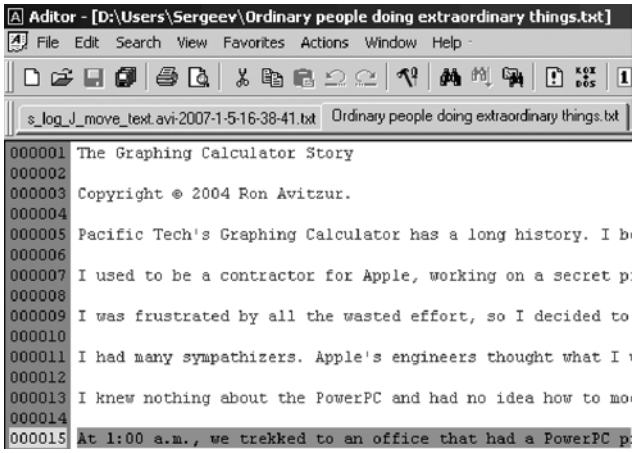
Возьмем, например (рис. 4, *а*) большое количество цветного текста. Весь этот текст можно свободно прочитать на всех трех компонентах YCrCb (рис. 4, *б*—*г*), что, как говорилось ранее, является признаком сильной избыточности.

В работе предлагается некоторая модификация алгоритма предварительной обработки, расширение YUV(YCrCb)-преобразования (Extended YUV, E-YUV), которое поможет избавиться от подобной избыточности и таким образом повысить эффективность работы кодеков.

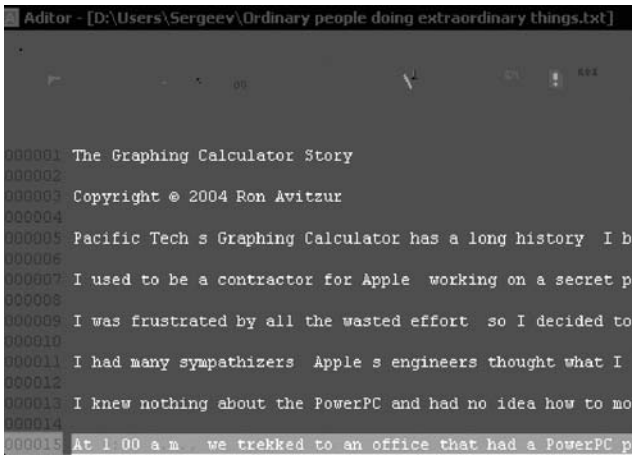
а)



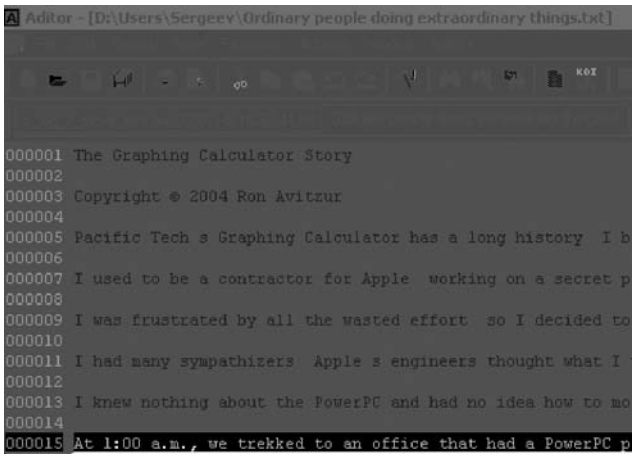
б)



в)



г)



■ ***Рис. 4.*** *Цветной текст: а — исходное изображение; b — компонента Y; в, г — компоненты Cr и Cb соответственно*

Расширение YUV: E-YUV

Если внимательно посмотреть на преобразование YCbCr, то можно заметить следующий факт: существуют две точки — набор для черного цвета (RGB = = 0, 0, 0) и набор для белого цвета (RGB = 255, 255, 255), для которых можно однозначно определить по одному набору YCbCr (черный YCbCr = = 0, 0, 0 и белый YCbCr = 255, 128, 128). Такие значения Y (0 и 255) возникают только для таких наборов RGB. Отсюда можно вывести правило 1:

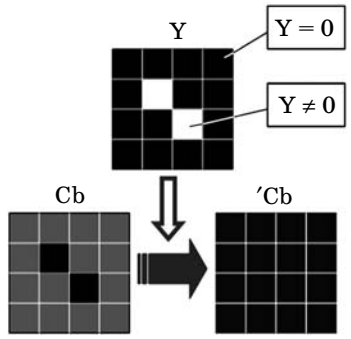
- если RGB = 0, 0, 0, то YCbCr = 0, 0, 0;
- если Y = 0, то RGB = 0, 0, 0,

т. е. при обратном переопределении значения компонент CbCr совсем не участвуют, и если мы даже подменим значения внутри этих компонент, черный цвет все равно будет однозначно восстановлен. Следовательно, если какое-либо изображение встретится на черном фоне (YCbCr = 0, 0, 0), то возможно в компонентах CbCr заменить все полученные нули на такие значения, которые сделают вид этих компонент более гладким, менее зависимым и более удобным для сжатия. «Фоновым» является цвет с наибольшей частотой повторения.

Правило 2:

- если RGB = 0, 0, 0, то Y = 0, а в Cb и Cr можно записать любые значения.

На рис. 5 приведен пример работы правила 2, где «фоновый» черный цвет в компоненте Cb заме-



■ ***Рис. 5.*** *Пример замены значений в компоненте Cb*

няется значениями, которые были записаны в нефоновой части изображения (домена).

Правила 1 и 2 будут работать только для тех изображений, где присутствует черный цвет. Для цветных фонов правило 2 напрямую не применимо по двум основным причинам. Во-первых, нет однозначности при обратном определении, и по значению Y нельзя сказать, какие точно были исходные RGB. Например, значение Y = 119 может быть получено из 120 тыс. различных наборов RGB (рис. 6, см. с. 3 обложки).

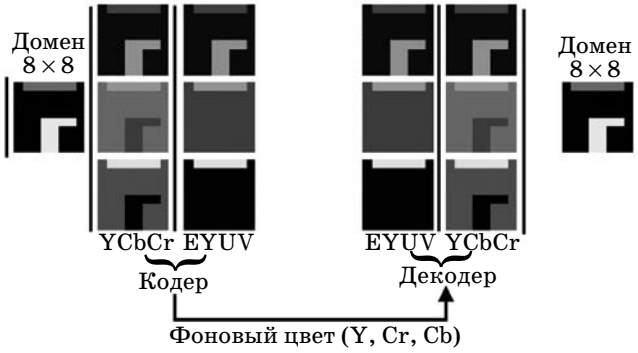
Это означает, что если в нашем изображении встретятся два набора RGB, у которых значения Y совпадут, и если один из этих цветов являлся фоновым, то соответственно CbCr-компоненты этого цвета будут переопределены. При обратном преобразовании второй цвет с совпавшим Y будет заменен на фоновый цвет, что приведет к явным искажениям. К счастью, вероятность возникновения такого события при размере домена 8×8 пикселей крайне мала, но для абсолютной надежности домены с такими ситуациями можно пропускать, т. е. не проводить переопределение. Таким образом, правило 2 примет следующий вид:

• если в домене значение Y′, присущее фоновому цвету RGB, не совпадает ни с одним Y из всех присутствующих цветов в этом домене, то тогда CbCr-компоненты в точках, где компонента Y равна Y′, можно переопределить.

Во-вторых, в случае цветного фона декодеру необходима информация о том, какой цвет считать фоном для обратного переопределения. Очевидно, эти данные (3 цифры на домен) нужно передавать в виде служебной информации.

Пример переопределения компонент CbCr с использованием правила 2 для цветных изображений проиллюстрирован на рис. 7.

После такой модификации правило 2 становится применимо для всех возможных изображений,

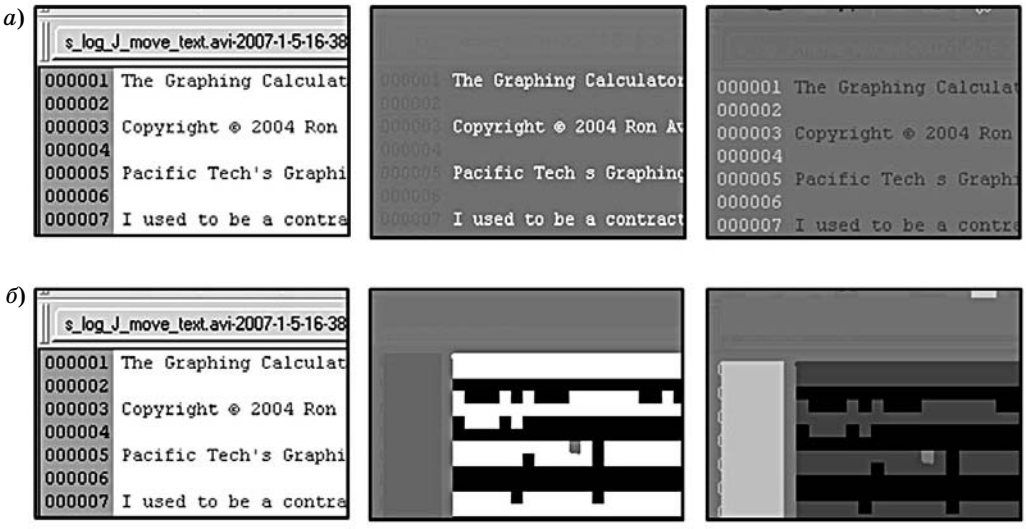


■ ***Рис. 7.*** *Пример работы правила 2 для цветных доменов*

но остаются следующие вопросы: какой цвет считать фоновым и какими значениями переписывать компоненты Cb и Cr.

Эти вопросы можно решить следующим образом: *фоновым* является цвет, который наиболее часто повторился в домене. Значения Cb и Cr будут браться из цвета, второго по частоте. Доказательство оптимальности такого переопределения компонент Cb и Cr в данной статье не приводится. Однако очевидно, что оно уменьшает общее количество различных значений в компонентах Cb и Cr, что должно приводить к уменьшению энтропии, а следовательно, и к увеличению степени сжатия.

Открытым вопросом во всем алгоритме является способность заранее отличить домены, на которых применение вышеописанных действий приведет к выигрышу по степени сжатия или только к увеличению дополнительной информации, так как, например, черно-белый текст хорошо сжимается и без дополнительного преобразования. На текущий момент авторами применяется эмпирически полученная формула, основанная на частотах цветов в домене:



■ ***Рис. 8.*** *Вид компонент Y, Cb и Cr после применения стандартного (а) и расширенного (б) преобразования*

■ ***Таблица 2.*** *Результаты сравнения YUV и E-YUV-преобразований при использовании кодеков H.264/AVC и JPEG-LS*

Изображение	H.264/AVC			JPEG-LS			Характеристика изображения
	YUV: размер после сжатия, байт	E-YUV: размер после сжатия, байт	Выигрыш E-YUV против YUV, %	YUV: размер после сжатия, байт	E-YUV: размер после сжатия, байт	Выигрыш E-YUV против YUV, %	
Photo	3030602	3030602	0,00	2909749	2909749	**0,00**	Фото-изображения
Lena	395712	395712	0,00	375456	375456	**0,00**	
Peppers	434445	434445	0,00	411528	411528	**0,00**	
desc_movea	622167	617344	0,78	501858	493248	**1,72**	Снимки с рабочего стола при работе с различными типами программ
a_test005	1050993	1041751	0,89	796403	787184	**1,16**	
a_test004	1155755	1143220	1,10	923348	904127	**2,08**	
a_test007	1122772	1110026	1,15	1183964	1103286	**6,81**	
a_test003	1288251	1269895	1,45	1176360	1149495	**2,28**	
visio_plus0	310398	301517	2,86	248261	244845	**1,38**	
opera_plus0	366183	353466	3,47	288959	287754	**0,42**	
a_test001	365889	353031	3,64	295837	287983	**2,65**	
6_text	574360	538562	6,23	180070	146962	**18,39**	
0_text	201917	188771	6,51	197270	183318	**7,60**	
VisualStudio	899765	619081	31,22	405785	285006	**29,76**	Снимки при работе в текстовых редакторах
5_text	986067	528283	46,43	545527	302538	**44,54**	
Calendar	1087062	518678	52,31	608070	318678	**47,62**	

$$\textit{Если}\left(\max_i(C_i) > \left(\sum_{i=1}^{n} C_i - \frac{\sum_{i=1}^{n} C_i}{n}\right)\right),$$

тогда переопределять, иначе нет,

где C_i — количество повторений набора i; n — количество различных цветов.

Можно наблюдать компоненты Y, Cb и Cr после стандартного преобразования и после работы расширенного цветового преобразования (рис. 8).

Дополнительными данными в этом преобразовании будут значения RGB для фонового цвета. Так как все обратное переопределение основывается на компоненте Y, то значение фонового Y можно свободно вычислить из значений RGB, переданных на сторону декодера. Размер этих данных при работе с доменами размером 8×8 составит 1/64 от исходных данных. По своей сути эти дополнительные данные представляют собой уменьшенное в 64 раза исходное изображение, в котором осталась лишь информация о присутствовавших фонах. Так как эти данные имеют некоторые закономерности, то их можно подвергнуть дополнительному сжатию, используя кодер длин серий [10] с добавлением монотонного кода Галлагера—ван Вухриса [11] (некоторая упрощенная реализация JPEG-LS). Применение этого кодера для дополнительных данных дает сжатие в среднем в 3–4 раза.

Результаты применения E-YUV для сжатия изображений

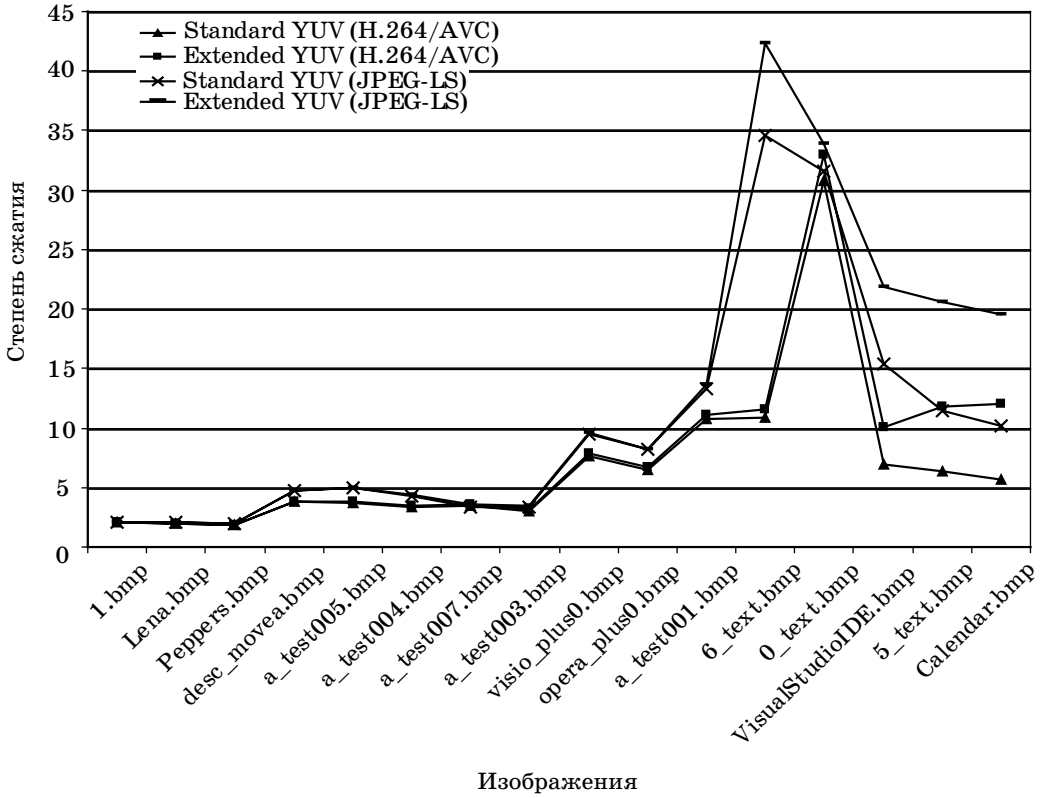
Для оценки предложенного алгоритма используются результаты, полученные после кодирования обычного преобразования YCbCr и расширенного преобразования E-YUV кодером H.264/AVC в режиме без потерь и кодером JPEG-LS [12] (табл. 2).

Изображения 1.bmp, Lena.bmp и Peppers.bmp — это фотореалистичные изображения, и они имеют степень сжатия точно такую же, как и при стандартном преобразовании, так как детектор не определил в них наличия доменов, на которых можно было бы получить выигрыш. Дополнительный поток при этом состоит из нулей (ноль — признак того, что E-YUV не применялось) и может быть эффективно закодирован методами, описанными выше.

Изображения с desc_movea.bmp до o_text.bmp представляют собой снимки с экрана при обычной работе на компьютере, когда в равной степени присутствуют и текст, и фотографии.

Примеры от VisualStudioIDE.bmp до Calendar.bmp — это также снимки с экрана, но основным содержимым этих изображений является цветной текст в различных текстовых редакторах.

Итоговый график сравнения расширенного и стандартного преобразования показан на рис. 9.



■ ***Рис. 9.*** *Сводный график использования стандартного YCbCr и E-YUV*

Заключение

Предлагаемое в статье расширенное цветовое преобразование E-YUV позволяет более эффективно сжимать синтетические изображения и изображения, содержащие компьютерную графику. Эксперименты с использованием кодеков JPEG-LS и H.264/AVC показали, что выигрыш при сжатии без потерь достигает 52 %. Эффективность сжатия фотореалистичных изображений не меняется. Предлагаемый подход обладает низкой степенью сложности.

Работа выполнена при поддержке гранта Intel CTG Research Council.

Литература

1. **Wiegand T., Sullivan G. J., Bjontegaard G., Luthra A.** Overview of the H.264/AVC Video Coding Standard // IEEE Transactions on Circuits and Systems for Video Technology. July, 2003. Vol. 13. N 7. P. 560–576.
2. **JPEG2000** requirements and profiles version 6.3, Coding of Still Pictures, ISO/IEC JTC 1/SC 29/WG 1, N1803. July, 2000.
3. **Красильников Н. Н.** Цифровая обработка изображений. М.: Вуз. книга, 2001, 319 с.
4. http://en.wikipedia.org/wiki/RGB_color_model
5. http://en.wikipedia.org/wiki/CMYK
6. http://en.wikipedia.org/wiki/HSV_color_space
7. http://en.wikipedia.org/wiki/YPbPr
8. **Van Rijsselberger D.** YCoCg(-R) Color Space Conversion on the GPU. Sixth FirW PhD Symposium, Faculty of Engineering. Belgium: Ghent University Press, 30th November 2005. P. 102.
9. **CIE Colorimetry.** Official recommendations of the International Commission on Illumination. Vienna, Austria: Bureau Central de la CIE, 1986.
10. **Golomb S. W.** Run-length encodings // IEEE Trans. Inform. Theory. July, 1966. Vol. IT-12. P. 399–401.
11. **Gallager R., van Voorhis D.** Optimal source codes for geometrically distributed alphabets // IEEE Trans. Inform. Theory. 1975. Vol. 21 (2). P. 228–230.
12. **Lossless** and near-lossless coding of continuous tone still images (JPEG-LS): Final Committee Draft #14495, FCD 14495, 1997.