УДК 004.932.4

С. З. Свердлов, канд. тех. наук, профессор кафедры прикладной математики, Вологодский государственный университет

Алгоритм и программа для уменьшения цифровых изображений

Аннотация. *Рассматриваются алгоритм и реализующая его программа, предназначенные для уменьшения растровых цифровых изображений. За счет использования колориметрически-корректных преобразований и эффективного метода масштабирования, имеющего естественные основания, обеспечивается высокое качество масштабирования: правильная передача тонов, корректная цветопередача, высокая детализация.*

Ключевые слова: обработка изображений, цифровая фотография, масштабирование, растровое изображение, программное обеспечение, колориметрия.

Введение

Средства масштабирования цифровых растровых изображений широко распространены. В частности, востребованным является уменьшение изображений перед их публикацией в интернете. В работе представлен эффективный алгоритм масштабирования и реализующая его программа, которые обеспечивают высококачественное уменьшение изображений. Приняты следующие подходы:

1. Преобразования изображения выполняются в линейном цветовом пространстве. В противном случае при вычислении цвета пикселей будут нарушены фундаментальные законы физики и колориметрии, в частности, закон аддитивности Г. Грассмана: цветовые координаты суммы излучений равны сумме координат составляющих [1], [2]. В задаче уменьшения изображений это особенно актуально, поскольку цвет пикселей результирующего (уменьшенного) изображения формируется как смесь цветов исходных пикселей.

2. Промежуточные вычисления выполняются в вещественных, а не в целых числах. Это обеспечивает отсутствие ошибок округления. В рассматриваемом случае это важно, в частности, потому, что при линеаризации исходного гамма-корректированного изображения значения RGB-компонент уменьшаются, что при использовании целочисленного представления привело бы к существенным потерям.

3. Использованный алгоритм масштабирования базируется на естественных основаниях и относится к категории алгоритмов, основанных на площадях. Идея этого алгоритма известна [3], [4]. Предлагаемая автором его эффективная реализация, использующая вещественное представление цветовых линеаризованных координат, позволяет получить результаты, которые зачастую дают лучшее качество масштабирования, чем широко используемые методы.

4. После масштабирования предусматривается регулируемое усиление резкости и выравнивание контраста.

Линеаризация

Переход к линейным цветовым координатам выполняется по достаточно сложным в вычислительном отношении формулам. Например, для цветового пространства sRGB [5] они выглядят так:

$$C_{лин} = \begin{cases} \dfrac{C_{srgb}}{12.92}, & C_{srgb} \le 0.04045 \\ \left(\dfrac{C_{srgb}}{1{,}055}\right)^{2.4}, & C_{srgb} > 0.04045 \end{cases} \qquad (1)$$

где C_{srgb} — исходные гамма-корректированные sRGB-значения цветовых координат R, G или B (формулы для преобразования R, G и B одинаковы); $C_{лин}$ — линеаризованные значения R, G или B. Предполагается, что цветовые координаты выражаются вещественными числами в диапазоне от 0,0 до 1,0.

Как видно, вычисления по формуле (1) связаны с возведением в дробную степень, что требует существенных затрат. Поэтому преобразование, подобное (1), табулируется. При использовании 8-битного представления изображения

количество возможных значений C_{srgb} равно 256, в случае 16-битного представления — не более 65536. Ни по затратам памяти, ни по быстродействию использование таблицы такого размера не создает проблем при доступных на современных компьютерах ресурсах.

Одновременно с преобразованием линеаризации выполняется и переход к вещественному представлению данных. Кроме этого, при табулировании преобразования (1) применяются нормирующие множители, которые используются в алгоритме масштабирования.

Таким образом, преобразования, которые должны быть применены к каждому пикселю, выполняются в ходе обработки одним обращением к таблице, требующим минимальных затрат.

Масштабирование

На рисунке 1 показана схема, иллюстрирующая принцип работы алгоритма масштабирования.

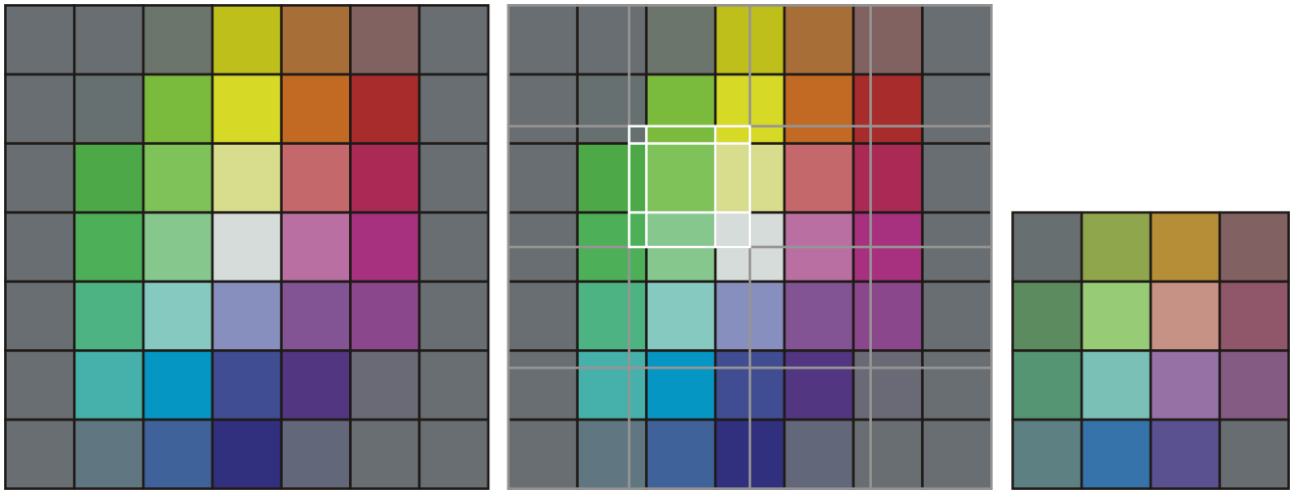

Рис. 1. Принцип масштабирования
Слева направо: исходное изображение 7x7; масштабирование до 4x4; результат

Рассмотрим масштабирование, при котором сохраняются пропорции изображения. Для иллюстрации возьмем картинку квадратного формата, что не уменьшает общности рассмотрения. Пусть есть растровое изображение $N \times N$ пикселей. На рис. 1 $N = 7$. Его нужно уменьшить до размера $n \times n$, $n < N$. В примере взято $n = 4$. Будем считать, что пиксели изображения — цветные квадра-

ты. Естественный способ получения изображения размером $n \times n$ из картинки $N \times N$ можно представить как фотосъемку или сканирование исходного изображения с помощью цветочувствительной матрицы размером $n \times n$, сенсоры которой (сенсели) также представляют собой квадраты. Цвет, зарегистрированный каждым сенселем, формируется как цвет суммы излучений тех частей исходных пикселей, которые покрываются данным сенселем. Цветовые координаты этой суммы вычисляются как сумма соответствующих цветовых координат пикселей, покрываемых данным сенселем, с весовыми коэффициентами, пропорциональными части площади пикселя, покрываемой сенселем. На рис. 1 светлым контуром выделены те части пикселей, которые участвуют в формировании цвета одного из сенселей.

Реализация масштабирования

Рассмотренный метод масштабирования допускает эффективную реализацию. Алгоритм обладает свойством сепарабельности. То есть обработка матрицы изображения может быть сведена к масштабированию сначала по строкам, а затем по столбцам (или наоборот): решение двумерной задачи сводится к решению двух одномерных. В связи с этим в дальнейшем будем рассматривать одномерный случай. На рисунке 2 показана строка из $N = 7$ пикселей, которая должна быть масштабирована до размера $n = 4$ пикселя (сенселя).

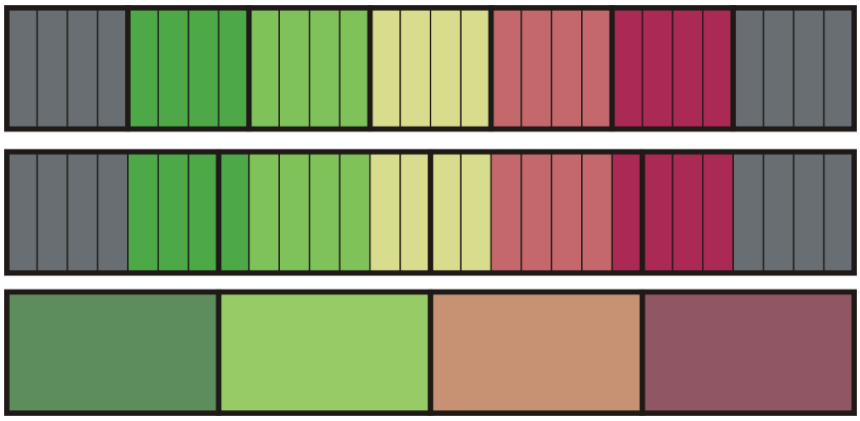

Рис. 2. Масштабирование строки
Сверху вниз: элементарные части в исходной строке; элементарные части в результирующей строке; цвет результирующих пикселей

Разобьем исходную строку ($N = 7$) и строку, которую нужно получить ($n = 4$) на $N \cdot n$ частей. В примере это $7 \cdot 4 = 28$ частей. Тогда и в каждый пиксель и в каждый сенсель укладывается целое число таких элементарных частей. В пиксель — n частей, а в сенсель — N (см. рис. 2). Масштабирование можно представить как распределение элементарных частей пикселей по сенселям. При этом RGB-координаты сенселей вычисляются как сумма RGB-координат элементарных частей, а RGB-координаты каждой элементарной части берутся равными $1/n$-й RGB-координат соответствующего пикселя. Поскольку, в отличие от показанной на рис. 2 схемы, действительный размер пикселя итогового изображения (сенселя) такой же, что и пикселя исходного изображения, получаемая сумма должна дополнительно масштабироваться множителем n/N (для одномерного случая).

Пусть значение G-координаты (зеленая компонента цвета) первого слева на рис. 2 пикселя исходной строки равно 28, а второго слева пикселя — 84 (имеются в виду линейные координаты из диапазона 0–255). На каждую элементарную часть пикселя приходится 1/4 часть значения координаты. То есть элементарная часть первого пикселя несет 28/4 = 7 «единиц зеленого», а второго — 84/4 = 21 «единицу зеленого». Цвет первого слева сенселя (рис. 2) образуется из четырех элементарных частей первого пикселя и трех частей второго. Значение G для первого сенселя вычисляется так: $G = (7 \cdot 4 + 3 \cdot 21) \cdot 4/7 = 52$. Нетрудно заметить, что вычисление можно сократить и воспользоваться формулой:

$$C = \frac{1}{N} \sum_{i=1}^{k} C_i m_i \,, \qquad (2)$$

где C – вычисленная цветовая координата сенселя; k – количество пикселей, влияющих на цвет сенселя; C_i – цветовая координата i-го пикселя; m_i – количество элементарных частей i-го пикселя, формирующих цвет сенселя.

Программную реализацию масштабирования рассмотрим для одномерного случая и монохромного изображения. Обработка трех цветовых каналов (R, G и

B) выполняется аналогично. Обработка каждой строки матрицы изображения выполняется одинаково. А обработка столбцов — аналогично обработке строк.

Каждый пиксель исходной строки может участвовать в формировании цвета одного или двух пикселей результирующей строки (сенселей). В первом случае он целиком «поглощается» одним сенселем, во втором — его значение делится между двумя соседними.

Обозначим `x[N]` – массив цветовых координат строки исходного (монохромного) изображения; `y[n]` – массив результирующей строки; `i` – индекс элемента исходной строки; `k` – индекс элемента результирующей строки. Перед формированием результирующей строки запишем в ее элементы нулевые значения:

```
for( int k = 0; k<n; k++ )
   y[k] = 0;
```

Рассмотрим каждый пиксель исходной строки. Он «состоит» из *n* элементарных частей, которые нужно распределить по пикселям результирующей строки (сенселям). Обозначим `d` — разность между количеством «свободного места» в очередном, `k`-м сенселе и значением *n* — количеством элементарных частей, которые надо разместить. Первоначально `k = 0`; `d = N - n`.

Можно выделить три случая:

1. Разность `d>0`. Свободного места более чем достаточно. К значению сенселя добавляется *n* элементарных частей (цветовая координата `i`-го пикселя умножается на весовой коэффициент равный *n*). Разность `d` уменьшается на *n*.
2. Свободного места ровно *n* элементарных ячеек, `d = 0`. К значению сенселя добавляется *n* частей, сенсель заполнен, готовится заполнение следущего сенселя (`k` увеличивается на единицу). Значение `d` становится таким же, как в самом начале.

3. Количество свободного места недостаточно для размещения *n* частей: `d < 0`. В этом случае *n* частей разделяются между текущим и следующим сенселями.

Получается такая программа:

```
d = N - n;
k = 0;
for( i = 0; i<N; i++ )
   if( d > 0 ) {
      y[k] += x[i]*n;
      d -= n;
   }
   else if( d == 0 ) {
      y[k++] += x[i]*n;
      d = N - n;
   }
   else {
      y[k] += (n+d)*x[i];
      y[++k] -= d*x[i];
      d += N - n;
   };
```

Эта программа вычисляет для каждого сенселя сумму, присутствующую в правой части формулы (2). Однако для получения значений цветовых координат сенселей требуется еще в соответствии с (2) применить множитель $1/N$. Отдельных затрат для этого не требуется, поскольку такую операцию можно совместить с линеаризацией исходных данных, которая выполняется по таблице, вычисленной заранее. Затраты при этом переносятся с момента обработки изображения на момент предварительного формирования таблицы, которая обеспечивает не только линеаризацию, но также учет множителя и переход от целочисленного представления цветовых координат к вещественному.

Свойства алгоритма масштабирования

Как видно из рассмотрения принципа масштабирования (рис. 1), в формировании результирующего изображения всегда участвуют все пиксели исходного. Это гарантирует сохранение детализации на итоговой картинке в той мере, насколько это возможно при уменьшении размера. При билинейной интерполяции используются 4 пикселя исходного изображения для вычисления цвета

одного результирующего, при бикубической — 16 пикселей. При уменьшении более чем вдвое в случае билинейной интерполяции и более чем вчетверо в случае бикубической неизбежна потеря детализации. Но и при меньшей кратности уменьшения рассматриваемый алгоритм может использовать больше информации, чем билинейная и бикубическая интерполяция. Так, при кратности уменьшения от 1 до 2 количество пикселей исходного изображения, участвующих в формировании цвета одного результирующего пикселя, может быть равным 9, при масштабе от 2-х до 3-х — 16, при масштабе от 3 до 4 — 25.

При кратности уменьшения, не превышающей 2, алгоритм работает примерно с той же скоростью, что и билинейный. При большей кратности билинейный может работать быстрей, но при этом теряет данные.

К недостаткам алгоритма можно отнести некоторое снижение визуальной резкости итогового изображения.

Усиление резкости

После масштабирования выполняется регулируемое усиление резкости. Поскольку основным назначением уменьшенного изображения считается его использование для просмотра на экране в масштабе 1:1, оказывается возможным использование простого и эффективного алгоритма. Использование линейного цветового пространства для преобразования позволяет избежать артефактов, характерных для некоторых алгоритмов усиления резкости. Расчет для строки (для каждого из 3-х RGB-каналов) выполняется по формуле

$$y_i = \alpha x_i + (1-\alpha)\frac{x_{i-1} + x_i + x_{i+1}}{3};\ i \in [1, n-2], \qquad (3)$$

где α – сила эффекта; α > 1 соответствует усилению резкости; α < 1 — ослаблению. x_{i-1}, x_i, x_{i+1} —значения (цветовые координаты) трех очередных пикселей исходной (до усиления резкости) строки; y_i — значение i-го пикселя после преобразования. Обработка по столбцам выполняется аналогично.

Выравнивание контраста

В использованном алгоритме масштабирования вычисление цвета пикселей результирующего изображения выполняется усреднением (с весами) цвета нескольких пикселей исходного. При этом максимальное значение цветовых координат пикселей итогового изображения окажется не больше максимального значения цветовых координат исходного изображения. Если оно будет строго меньше, это будет означать уменьшение максимальной яркости масштабированного изображения. Аналогично обстоит дело и с минимальными значениями. То есть в результате масштабирования возможно уменьшение общего контраста изображения.

В связи с этим предусмотрено выравнивание контраста. Для этого в ходе обработки определяются минимальные и максимальные значения цветовых координат пикселей исходного изображения, а затем цветовые координаты уменьшенного изображения масштабируются так, чтобы минимальные и максимальные значения цветовых координат уменьшенного и исходного изображения совпадали.

Программа C3C Image Size

Рассмотренный алгоритм уменьшения изображений реализован в программе C3C Image Size (рис. 3) [6]. Программа представляет собой подключаемый модуль (плагин) для Adobe Photoshop. Существуют версии для ОС Windows и Mac OS (адаптацию для Mac OS выполнил А. Данильченко).

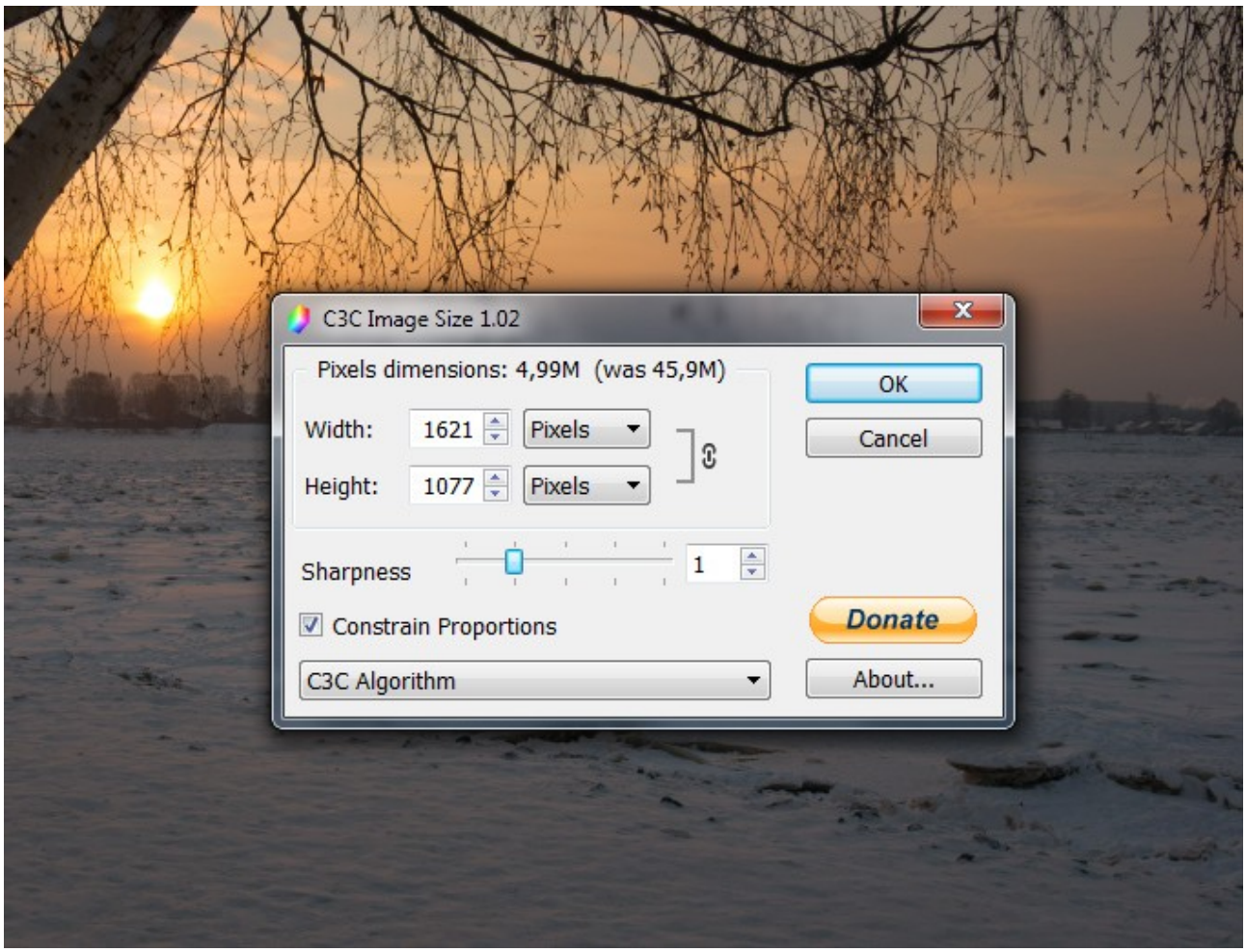

Рис. 3. Программа C3C Image Size. Версия для Windows

Параметр Sharpness (резкость) задается по логарифмической шкале. Значение Sharpness равное 0 соответствует отсутствию усиления резкости (α =1), значение 1 соответствует $\alpha = \sqrt{2}$.

Тестирование

Предъявление примеров масштабирования растровых изображений в печатном документе представляет определенные сложности. То, что мы видим на бумаге — результат работы программ и оборудования, используемых при печати. Эти программы в том числе выполняют и масштабирование. В силу этих обстоятельств приведенные ниже иллюстрации могут дать лишь приблизительное представление о работе обсуждаемых алгоритмов. Для корректного сопоставления нужно рассматривать оригиналы изображений, как исходных, так и

уменьшенных. Эти оригиналы доступны на сайте программы C3C Image Size [6].

Черно-белая мишень

Первый пример — масштабирование черно-белой мишени. Исходное изображение (рис. 4) имеет разрешение 1200x1200 пикселей. Оно масштабировалось до размера 200x200 программами C3C Image Size, Adobe Photoshop CC (выпуск 2015, алгоритм bicubic) и плагином Resize [7] к программе ImageJ 1.49v [] (алгоритм Least-Squares).

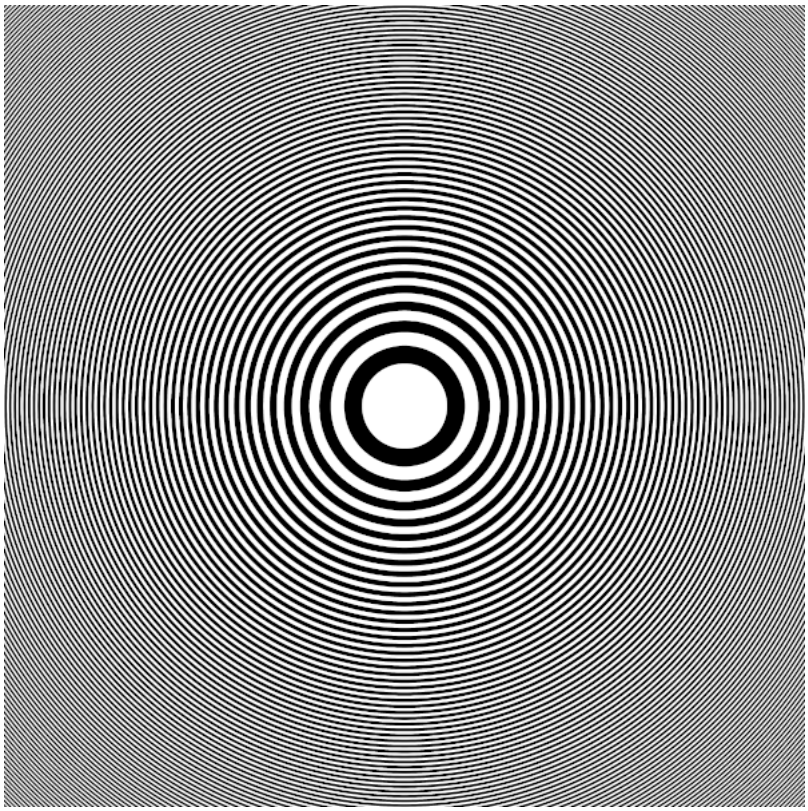

Рис. 4. Черно-белая мишень

Результаты масштабирования показаны на рис. 5 и рис. 6. Чтобы улучшить качество воспроизведения масштабированных изображений в печатном документе, после масштабирования выполнено увеличение вдвое с использованием алгоритма «ближайшего соседа», который в данном случае «учетверяет» каждый пиксель.

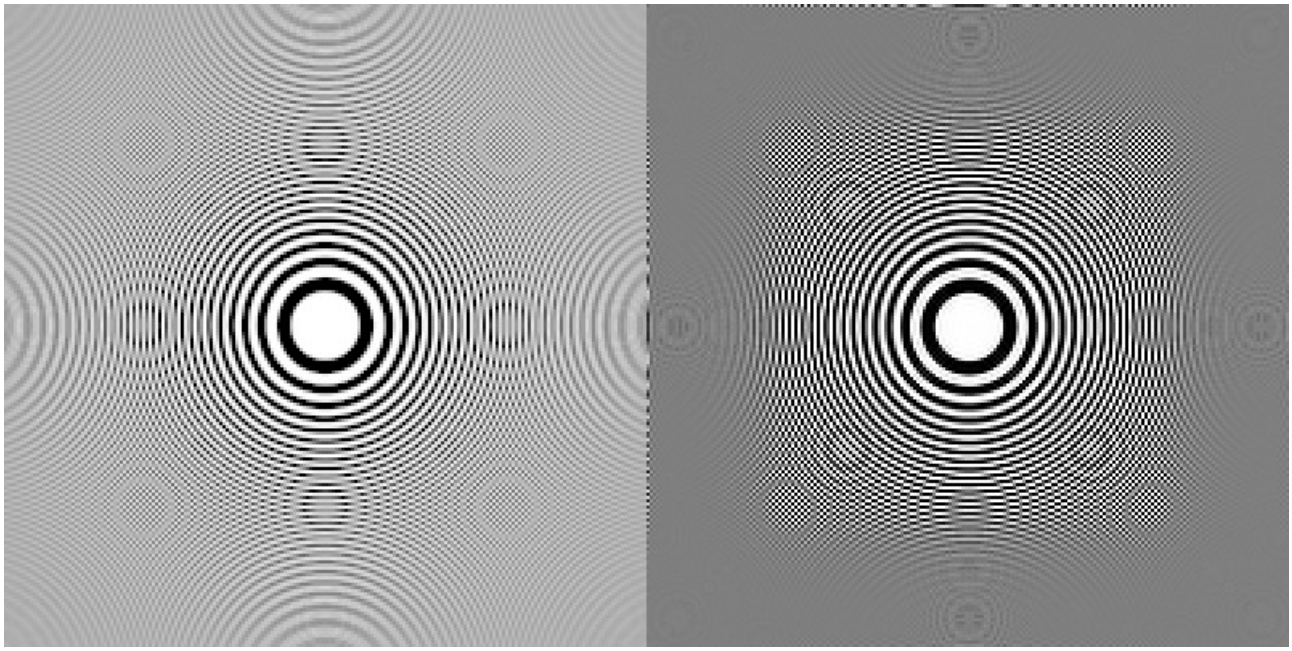

Рис. 5. Слева — результат C3C Image Size; справа — ImageJ

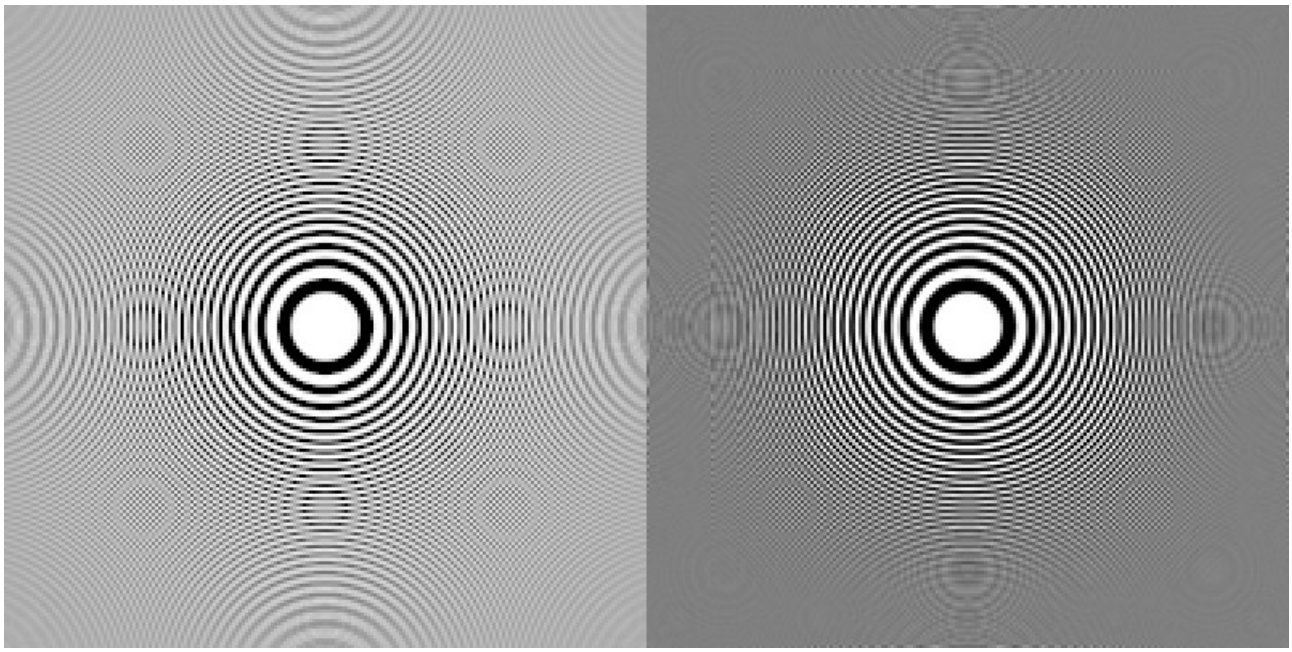

Рис. 6. Слева — результат C3C Image Size; справа — Adobe Photoshop

Обращает на себя внимание, что варианты ImageJ и Adobe Photoshop дают затемнение по периферии мишени. По эффекту муара варианты Adobe Photoshop и C3C Image Size выглядят предпочтительней варианта ImageJ.

Фотография

В качестве примера для масштабирования взят снимок «Русский Север» (рис. 7), сделанный в Великом Устюге в январе 2016 года. Исходный размер фотографии 4912x3264 пикселя — оригинальный размер, получаемый с фотокамеры Sony NEX-6.

Рис. 7. «Русский Север»

Фотография масштабировалась до «веб-размера» — 900 пикселей по длинной стороне (линейное уменьшение в 5,46 раза). Затем, чтобы можно было сравнивать результаты масштабирования с исходным изображением, а также для того, чтобы эти результаты лучше воспроизводились в печатном документе, уменьшенные изображения были снова приведены к исходному размеру с помощью алгоритма «ближайшего соседа». Фрагменты этих изображений показаны на рисунках 8–11.

Рис. 8. Фрагмент исходного изображения

Рис. 9. Масштабирование с помощью программы C3C Image Size

Рис. 10. Масштабирование с помощью ImageJ Resize

Рис. 11. Масштабирование с помощью программы Adobe Photoshop (bicubic)

Сравнивая детализацию разных вариантов масштабирования, можно обратить внимание на следующие объекты на исходном снимке (рис. 8):

- В нижней части слева видны два человека, переходящих по льду замерзшую реку Сухону. Сказать уверенно, что на каком-либо из вариантов при уменьшении пара (то, что их двое) осталась хорошо различима, вряд ли можно. Но, по-моему, на рис. 9 что-то просматривается.

- На противоположном берегу стоит автомобиль с включенными фарами. Лучше всего наличие двух фар передано в варианте ImageJ (рис. 10). Две фары различимы в варианте C3C Image Size и совсем неразличимы в случае масштабирования с помощью Adobe Photoshop.
- На исходном фрагменте хорошо видны пять крестов на церквях Дмитрия Солунского и Сергия Радонежского. При масштабировании программой C3C Image Size остаются различимы два креста; программой ImageJ — один; Adobe Photoshop — ни одного.
- Можно также обратить внимание на передачу деталей в изображении большого дома, расположенного справа от церквей. Предпочтительным представляется вариант C3C Image Size.

Специальный тест

Ниже приведены результаты испытания программ на тестовом изображении (рис. 12), специально созданном для того, чтобы продемонстрировать некорректное масштабирование программами, которые используют для преобразований нелинейное цветовое пространство.

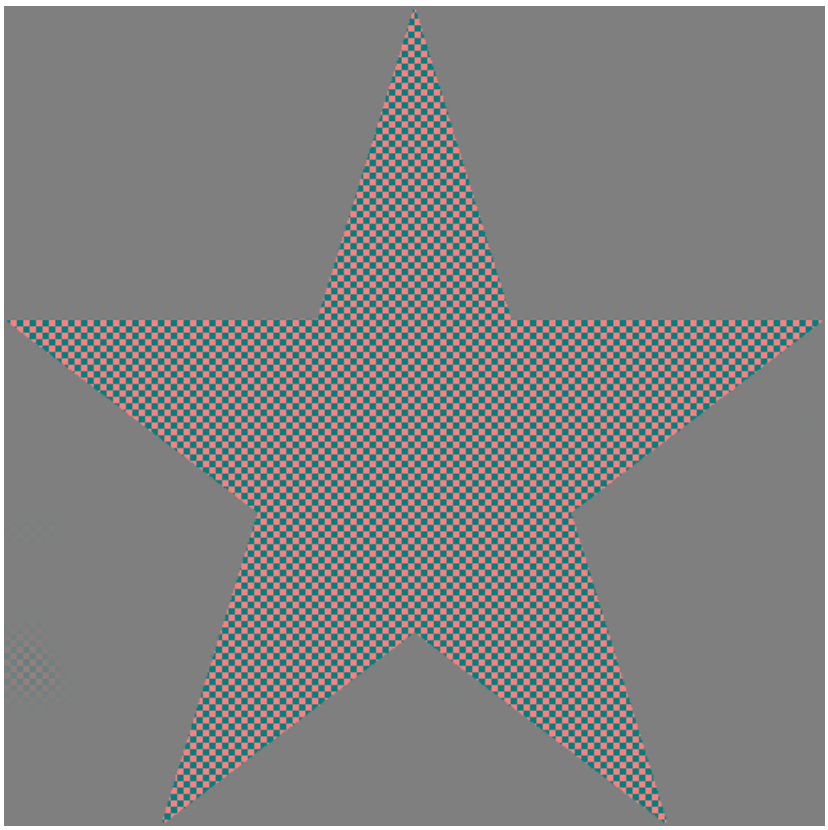

Рис. 12. Тестовое изображение

На рис. 13 показан увеличенный фрагмент тестового изображения. В оригинале цветные квадраты имеют размер 4x4 пикселя, а общий размер изображения 512x512.

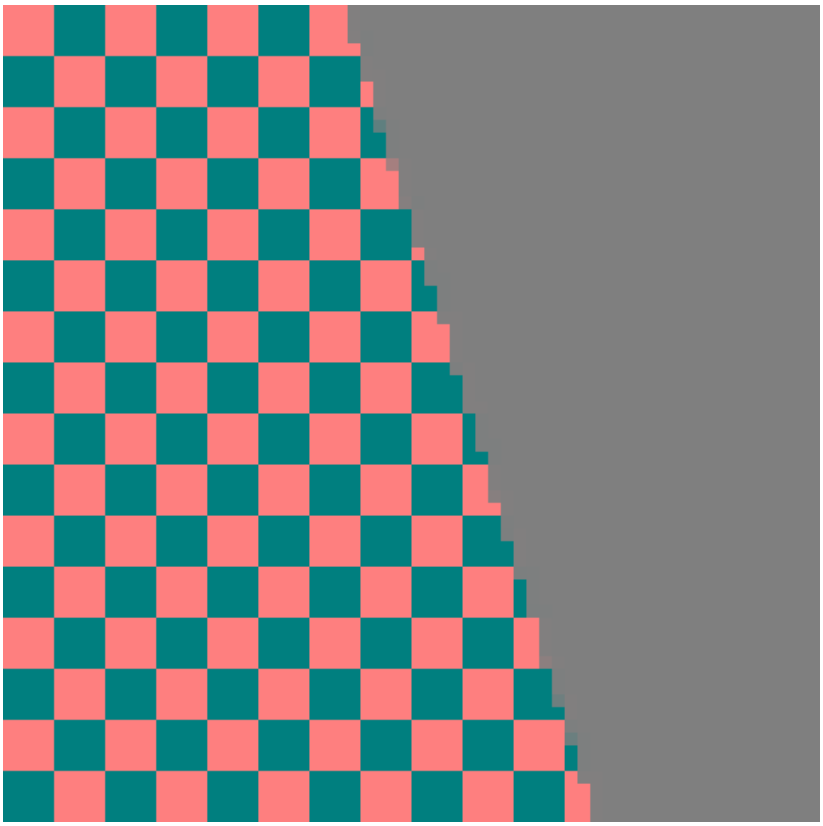

Рис. 13. Увеличенный фрагмент тестового изображения

Прежде чем обсуждать результаты тестирования, сформулируем критерий правильного преобразования: если преобразование, выполняемое с изображением, не имеет целью получение каких-либо специальных эффектов, то результат трансформации должен совпадать с тем, что видит человек в тех обстоятельствах, которые соответствуют такой трансформации. В нашем случае, когда речь идет об уменьшении изображения, результирующее изображение должно выглядеть так же, как исходное изображение, которое наблюдатель рассматривает с большего расстояния. Например, уменьшенная вдвое картинка должна выглядеть при обычном расстоянии просмотра так же, как исходная с расстояния, превышающего обычное расстояние просмотра в два раза.

Если рассматривать изображение на рис. 12 издалека, мы всегда, пока видимый размер изображения не достигнет предела разрешающей способности глаза, видим звезду бледно-красного цвета на сером фоне. Соответственно, при масштабировании этого изображения на уменьшенном варианте должна быть видна такая же звезда. Но это оказывается не всегда так.

При тестировании исходное изображение уменьшалось в 8 раз до размера 64x64 пикселя. Затем, для того чтобы результат можно было представить в печатной публикации, результирующие изображения были увеличены в 8 раз с помощью алгоритма «ближайшего соседа». Результат показан на рис. 14 (масштабы на рис. 12 и рис. 14 различны).

Рис. 14. Результат масштабирования
Слева направо: C3C Image Size, ImageJ, Adobe Photoshop

Полученный результат, разумеется, обусловлен тем, что тестовое изображение специально построено таким образом, чтобы некорректное поведение программ, использующих нелинейное цветовое пространство для преобразований, было очевидно. На результат масштабирования реальных фотографий, подобное некорректное поведение тоже влияет, но это влияние не так бросается в глаза.

Заключение

Представленный алгоритм и реализующая его программа C3C Image Size обеспечивают высококачественное уменьшение растровых изображений. Программа также предоставляет возможность регулировки резкости. Это помогает компенсировать возможный недостаток резкости уменьшенного изображения. Использование линейного цветового пространства гарантирует корректное вычисление цветов уменьшенного изображения. Тесты демонстрируют правильную тонопередачу без затемнения или осветления участков изображения, корректную цветопередачу, высокую детализацию уменьшенного изображения. Использованный алгоритм допускает эффективную реализацию.

Программа C3C Image Size [6] получила распространение среди фотографов и доступна для свободного использования.

Литература

1. Wyszecki G., Stiles W. S. Color Science. Concepts and Methods, Quantitative Data and Formulae. Second Edition, Wiley-Interscience Publication, 2000.
2. Юстова Е. Н. Цветовые измерения (Колориметрия). – Спб.: Издательство С.-Петербургского университета, 2000. – 397 с. ISBN 5-288-02648-3
3. Chun-Ho Kim, et al., Winscale: An Image-Scaling Algorithm Using an Area Pixel Model, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 6, June 2003, 549-553.
4. Summers D. Pixel Mixing. URL: http://entropymine.com/imageworsener/pixelmixing
5. INTERNATIONAL STANDARD IEC 61966-2-1 ed1.0. Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management – Default RGB colour space – sRGB.
6. С. Свердлов. C3C Image Size. Высококачественное уменьшение изображений [электронный ресурс]. URL: http://www.uni-vologda.ac.ru/~c3c/plug-ins/c3cimagesize.htm
7. An ImageJ plugin to resize an image using high-quality interpolation. URL: http://bigwww.epfl.ch/algorithms/ijplugins/resize/
8. ImageJ. Image Processing and Analysis in Java. URL: http://imagej.nih.gov/ij/

S. Z. Sverdlov, c3c@uni-vologda.ac.ru, Vologda State University, Vologda, Russia

Algorithm and program to downsizing the digital images

Abstract

The scaling algorithm, which provides high-quality digital images downsizing, and C3C Image Size program, that implements it, are described. They are developed based on the following principles:

1. Images are transformed in a linear color space. Otherwise, the calculation of the pixels color would violate fundamental laws of physics and colorimetry, in particular, H. Grassmann's law of additivity: the color coordinates of the sum of radiations are equal to the sum of the components' coordinates. It's especially relevant for the images size reduction, because the color of pixels of the resulting (downsized) image is formed as a mixture of color of source pixels.

2. Intermediate calculations are performed in real rather than integers, that ensures the absence of rounding errors. As RGB component values are reduced during the linearization of original gamma-corrected image, using integer representation would lead to substantial loss of accuracy, that's why it's especially important to avoid rounding errors.

3. Scaling is performed using an algorithm, which has a natural physical interpretation.

4. Controlled sharpness gain and contrast leveling are applied after scaling.

Tests show proper transmission of tones without darken or lighten areas on the picture, correct color rendition, high quality of the reduced image. The algorithm allows effective implementation, that provides high operating speed of the program. C3C Image Size program has spread among photographers and is available for free use.

Keywords

Image processing, digital photo, resize, raster graphics, software, plug-in, colorimetry.

References

1. Wyszecki G., Stiles W. S. Color Science. Concepts and Methods, Quantitative Data and Formulae. Second Edition, Wiley-Interscience Publication, 2000.
2. Justova E. N. Cvetovye izmerenija (Kolorimetrija).(The color measurement (Colorimetry)) – S-Pb.: S-PbU Publishing, 2000. – 397 p., ISBN 5-288-02648-3
3. Chun-Ho Kim, et al., Winscale: An Image-Scaling Algorithm Using an Area Pixel Model, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 6, June 2003, 549-553.
4. Summers D. Pixel Mixing. http://entropymine.com/imageworsener/pixelmixing/
5. INTERNATIONAL STANDARD IEC 61966-2-1 ed1.0. Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management – Default RGB colour space – sRGB.
6. S. Sverdlov. S3S Image Size. Vysokokachestvennoe umen'shenie izobrazhenij: http://www.uni-vologda.ac.ru/~c3c/plug-ins/c3cimagesize.htm
7. An ImageJ plugin to resize an image using high-quality interpolation: http://bigwww.epfl.ch/algorithms/ijplugins/resize/
8. ImageJ. Image Processing and Analysis in Java: http://imagej.nih.gov/ij/