СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ МАСШТАБИРОВАНИЯ РАСТРОВОЙ ГРАФИКИ

Абрамова Оксана Федоровна, Инкин Алексей Николаевич, Иванов Артем Евгеньевич Волжский политехнический институт (филиал) ВолгГТУ,
г. Волжский, Россия

Аннотация

В статье рассмотрены алгоритмы масштабирования пиксельной графики, а также проведен анализ алгоритмов с целью выяснить, какой из методов отображения будет наилучшим при увеличении изображений. В статье рассматриваются, в основном, алгоритмы, разработанные специально для улучшения качества изображений с маленьким разрешением. Особое внимание уделено разбору алгоритма депикселизации Копфа-Лисчински. Алгоритм предназначен для преобразования 8-битных изображений. При этом он решает проблему масштабирования пикселей и конвертирует их в области с разной степенью сглаживания, которые строго разделены контурными линиями.

Ключевые слова: алгоритмы масштабирования, растровая графика, изображение, алгоритм депикселизации, Копф-Лисчински

COMPARATIVE ANALYSIS OF ALGORITHMS FOR SCALING RASTER GRAPHICS

Abramova Oksana F., Ivanov A. E., Inkin A. N.
Volzhskiy Polytechnical Institute, branch of the Volgograd State Technical University
Volzhskiy, Russia

Abstract: In that article we describe image scale algorithms and analysis those algorithms to find out which of them is the best in terms of scaling pixel images. The article considered mainly algorithms developed specifically to improve image quality with a small resolution. Special attention is given to the algorithm depictional (Kopf-Lischinski). The algorithm is designed to convert 8-bit images. He solves the problem of scaling of pixels and converts them in the field with different degrees of smoothing, which are strictly separated by contour lines.

Keywords: scale algorithms, pixel images, bitmap graphics, algorithm depictional, Kopf-Lischinski

Эпоха высоких технологий подарила нам блага добычи, обработки и хранения больших объемов информации. Человек так устроен, что визуальная информация воспринимается им легко, и она является чуть ли не основным источником знания. Картинки, образы и представления считается основным видом информации, вопросы отображения этой информации стояли, стоят и будут стоять на протяжении веков. А если говорить о вычислительной машине, то различные алгоритмы отображения графической информации - это основа представления информации в современном мире. Однако, если процесс отображения графики изучен и автоматизирован уже достаточно хорошо, то вопросы в области управления масштабированием изображения еще остаются, и поиск наилучших решений не закончен.

Растровая графика - это одна из форм представления цифрового изображения, которая создается на компьютере посредством специальных программ, в которых рисунок изменяется на уровне пикселей, а ширина и высота картинки настолько малы, что все точки отчетливо различимы. Достоинства растровой графики в том, что это наилучший выбор при сверхнизком разрешении и ограниченной палитре цветов, требует мало ресурсов памяти, благодаря использованию палитровых форматов с узким цветовым спектром, но несмотря на крайне низкую цветовую передачу растровое изображение не теряет своей выразительности. Однако, до сих пор существует некий набор проблем в формировании и использовании растровых изображений, наилучшее

решение которых еще не предложено. Например, проблема с масштабированием, существующие варианты решения которой и будут рассмотрены в этой статье.

Под масштабированием изображения следует понимать увеличение или уменьшение размера изображения с сохранением его пропорций. Этот прием часто используется в компьютерной графике, обработке видеофайлов, реализуется в видеотехнике. Масштабирование может производиться несколькими методами в зависимости от типа графики. Если работа ведется с векторной графикой, то проблем с масштабированием не происходит. Однако, при работе с растровым изображением нужно использовать специальные алгоритмы, которые и будут рассмотрены в данной статье. Растровая графика плохо масштабируема; при ресемплинге изображения рисунок приходится отрисовывать заново. Стандартные алгоритмы масштабирования, такие как билинейная и бикубическая интерполяция, использующиеся для фотографий, абсолютно не подходят для растрового рисунка — изображение становится размытым. Однако, существуют методы, увеличивающие чёткость изображений на больших разрешениях. Компьютеры способны выполнять эти алгоритмы в режиме реального времени.

Для начала хочется сказать о причинах появления методов изменения масштаба растровой графики. Масштаб рисунка можно изменить разными способами. Один из самых простых методов увеличения разрешения - алгоритм копирования самого ближнего пикселя, меняющий каждую точку 4-мя пикселями того же цвета, при этом рисунок, сохранив детали оригинала, покрывается “лестницей”. Этот же алгоритм применяется для более точных изменений масштаба, например, для масштабирования 99 % или 101 % убирая или копируя каждую сотую точку.

В настоящее время имеется два вида типовых алгоритмов изменения масштаба рисунка – билинейная и бикубическая интерполяция.

Билинейная интерполяция - это растяжение обычной интерполяции для функции двух переменных. Идея состоит в том, чтобы линейно интерполировать в одном направлении, затем стоит произвести те же действия, но в перпендикулярном направлении. Такой прием стал популярным в компьютерной графике, но, во время масштабирования цифровых изображений прослеживается высокая пикселизация изображения.

Данный тип интерполяция используется для вычисления цветов добавочных пикселей относительно основных, это дает возможность смягчать переходы. Значением функции в этом случае принимается расцветка пикселя, а квадрат, созданный 4 основными пикселями, считается единичным. У этого алгоритма есть и минусы, главный из которых состоит в том, что при увеличении в N раз картинки размером X на Y точек, в конце получится рисунок размером не NX на NY, а (N(X-1)+1) на (N(Y-1)+1) точек. А всё из-за того, что для последней точки начального изображения нельзя найти пару, с которой можно было бы провести интерполирование.

В бикубической же интерполяции происходит немного другой процесс, там происходит увеличение кубической интерполяции для функции 2-х переменных, значение которой заданы на 2D сетке. Результат кубической интерполяции — это гладкая функция, а не поверхность, которую получают во время билинейной интерполяции или интерполяции методом ближайшего соседа. Таким образом, бикубическая интерполяция зачастую применяют при обработке рисунков, позволяя получить более качественный рисунок относительно билинейной интерполяцией. Если говорить о бикубической интерполяции, то значение её функции в нужной точке получается через ее значения в 16 ближайших точках.

Так как координаты цветов текущей точки часто вычисляются методом интерполяции 4-ёх ближайших, полученный рисунок выходит размытым. Не смотря на то, что это приемлемо для цветных изображений, данный метод приводит к уменьшению контрастности (резкости на контурах), и, поэтому, этот алгоритм плох для изображений с индексированной палитрой. Следовательно, оба стандартных алгоритма, рассмотренных выше нам не подходят.

Есть еще один способ, который не упомянули - ***алгоритм ближайшего соседа.*** Принцип его работы таков для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования. Такой метод дает пикселизированное изображение при увеличении и сильно зернистое изображение при уменьшении. Метод прост, при этом изображение сохраняет резкость границ, однако проявляется алиасинг (в частности, диагонали похожи на «лесенку» из квадратов).

Следовательно, идеальному методу для ресемплинга растровой графики необходимо интерполировать зоны сплошного тона, сохранять резкость горизонтальных и вертикальных линий и смягчать (с помощью антиалиасинга) диагональные линии и кривые. Т. е. все рассмотренные типовые алгоритмы будут давать неверную первоначальную картинку. Следовательно, необходим такой алгоритм масштабирования растровых изображений, который сможет объединить несколько положительных эффектов от типовых алгоритмов, и сведет к минимуму их недостатки.

Было предпринято несколько попыток решения этой задачи. Но сначала хотелось бы сказать об эффективности алгоритма, ведь это самое важно требование. Так как основное место использования этих алгоритмов — это эмуляция старых консольных игр и игр под систему DOS, большинство из них предназначены для вывода динамического изображения в реальном времени (при условии маленького разрешения картинки на входе) и работают только при увеличении, кратном двум.

EPX

Обзор алгоритмов масштабирования растровой графики хотелось бы начать с алгоритма Eric'sPixeleXpansion (EPX) или *пиксельное увеличение Эрика*. Этот метод, был создан Эриком Джонстоном из LucasArts в 1992 году в процессе переноса ядра SCUMM с IBM PC (разрешение 320×200, 256 цветов) на первые цветные компьютеры компании Apple, с разрешением большим в два раза. Алгоритм можно описать следующим образом. Допустим, что дана матрица пикселей из 9 элементов и известен центральный и боковые пикселы, а угловые неизвестны. Тогда сравнивая боковые пикселы можно было получить угловые, при равенстве 3 пикселей, за угловые брался центральный пиксель.

Дальнейшие реализации этого метода (AdvMAME2x и Scale2x, созданные в 2001 году) имеют другую (более эффективную), но функционально идентичную, реализацию. Существуют еще пара алгоритмов AdvMAME4x/Scale4x — это двойной (два раза применённый) EPX. В свое время этот алгоритм считался очень эффективным и его часто использовали, но позднее появились еще более усовершенствованные алгоритмы.

Отдельно хочется отметить алгоритм Scale3x/AdvMAME3x. Этот алгоритм во многом схож с EPX, а главным отличием можно назвать то, что в нем оценивается не 4 пиксела, а все 9.

Eagle

Следующий метод ресемплинга растровой графики - Eagle. Принцип работы алгоритма: для всех точек на входе создается 4 выходных, изначально все точки окрашиваются в цвет текущего пикселя (как и в ближайшем соседе). Затем берутся точки сверху и слева, и, если они одного цвета (все три), то окрашиваем и левую верхнюю точку в этот цвет. Эти действия повторяются для всех 4 точек, и выполняется переход к следующему набору точек. Т.е., в итоге, например, единичная чёрная точка на белом фоне после этого метода исчезнет. Данная ошибка исправлена в методах 2xSaI и HQ3x.

Алгоритм депикселизации (Копфа-Лисчински)

На конференции SIGGRAPH в августе 2011 года сотрудник Microsoft Research Йоханнес Копф совместно с профессором Дани Лисчински представили исследовательскую работу с описанием нового алгоритма депикселизации.

Алгоритм предназначен для преобразования 8-битных изображений. При этом он решает проблему масштабирования пикселей и конвертирует их в области с разной степенью сглаживания, которые строго разделены контурными линиями. В оригинальном изображении соседние пиксели по диагонали связаны только 1 точкой, из-за этого мелкие детали при увеличении становятся визуально несвязанными. Это порождает неточности при определении связей диагональных соседей.

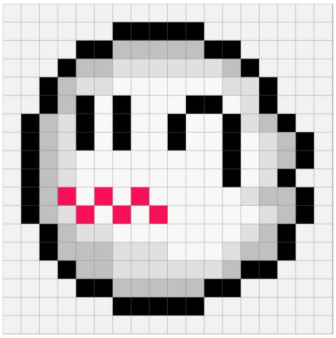
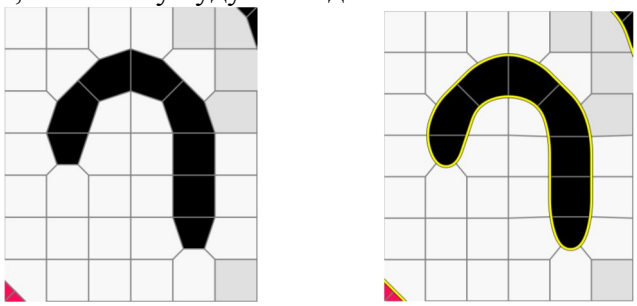


Рисунок 1 – исходное изображение 16x16 пикселей.

Основными примитивами в алгоритме являются B-сплайн кривые, которые определяют сглаживание контуров между областями. Рассмотрим граф с (w+1) (h+1) узлами, представляющий изображение размером WH. Каждый пиксель соответствует замкнутой ячейке графа. Горизонтальные и вертикальные соседние ячейки имеют соседние ребра в этом графике, а диагональные соседи разделяют только вершину. Эти диагональные соседи становятся визуально разделенными при масштабировании, в то время как соседи с общим краем остаются визуально связанными.

Первой задачей алгоритма является изменить исходные ячейки таким образом, чтобы каждая пара соседних пикселей, которые должны остаться соединёнными, соответствовала тем ячейкам, которые имеют общее ребро.

После изменения графа определятся грани, где соседние пиксели имеют существенно разные цвета. Эти грани мы помечаем как видимые, т.к. они формируют основу для видимых контуров в нашем окончательном векторном представлении, в то время как остальные ребра не будут непосредственно видимыми, поскольку будут находиться в сглаженных областях.



а) б)

Рисунок 2 а) реконструкция пикселей ячеек, отражающих свази в исходном графе; б) сплайны облегают видимые края.

Для получения сглаженных контуров квадратичные B-сплайн кривые вписываются в последовательности видимых рёбер. Это значительно улучшает плавность результата, однако изображение всё ещё страдает от ступенчатого эффекта. Поэтому в дальнейшем происходит оптимизация контрольных точек B-сплайн кривых.

Оптимизация происходит за счет поиска минимума суммы энергии каждого узла:

$$arg\ min_{p_i} \sum_i E^{(i)}$$

, где p_i – расположение i узла. Энергия узла определяется как сумма гладкости и позиции:

$$E^{(i)} = E_s^{(i)} + E_p^{(i)}$$

Эти два термина имеют равную долю в энергии. Гладкость определяется как отсутствие кривизны. Поэтому мы определяем энергию гладкости как:

$$E_s^{(i)} = \int_{s \epsilon r(i)} |k(s)|\, ds$$

, где r(i) – участок кривой, k(s) – кривизна в точке s. Интеграл вычисляется численно путём дискретизации кривой на фиксированном интервале.

Для того чтобы предотвратить слишком сильное изменение объектов, нужно дополнительно ограничить контрольные точки. Энергия позиции определяется как:

$$E_p^{(i)} = ||p_i - \hat{p}_i||^4$$

, где $\hat{p}_i$ – начальная позиция i узла. Возведение в четвертую степень позволяет узлам перемещаться относительно свободно в пределах их первоначальной позиции.

Тем не менее, из-за расположения кривых, контрольные точки сильно варьируются в связи с низким разрешением основной пиксельной сетки, результат всё ещё может быть ступенчатым. Поэтому на последнем этапе оптимизируются кривые линии, чтобы уменьшить ступенчатость.

Наконец отрисовывается изображение путём интерполяции цветов, с использованием радиальных базисных функций [5].

Рисунок 3 – результирующее изображение

Новый метод требует гораздо меньше ресурсов, делает обработку изображений в несколько раз быстрее. Алгоритм сработается с современными мониторами, которые имеют большую кадровую частоту. А это поможет поднять консоли и компьютерные дисплеи на абсолютно новый уровень графики.

Hqnx

Далее рассмотрим целое семейство алгоритмов hqnx, которые применяются для масштабирования. Максим Степин создал методы hq2x, hq3x и hq4x для ресемплинга в пропорциях 2:1, 3:1 и 4:1. Окрас каждой точки сравнивается с 8-ю соседями, последние отмечаются как ближние и дальние, после этого применяется прегенерированная таблица для нахождения нужного отношения значений для всех выходных точек (4, 9 или 16). Метод hq3x превосходно смягчает диагонали с наклоном ±1:2, ±1:1 и ±2:1 (если отсутствует антиалиасинг на входе метода); линии с другим множителем наклона представляются как ломаные из предыдущих линий. Отлично выравниваются крутые кривые. В отличие от 2xSaI, в выводе используется антиалиасинг. Хочется пояснить, что антиалиасинг это сглаживание — метод, используемый для устранения «лесенки», возникающей на краях изображений (плоских или объёмных) выводимых в один момент времени на монитор. Выравнивание придумали в 1972 в Массачусетском технологическом институте в ArchitectureMachineGroup, ставшей вдальнейшем основой MediaLab. Первоначально hqnx использовался для эмулятора SuperNintendo, ZSNES.

Напоследок хочется сказать о применении алгоритмов ресемплинга пиксельной графики в эмуляторах консолей. Быстрые компьютеры при применении этих методов дают возможность вывода изображения с другим масштабом со скоростью приложений реального времени, например, в играх. Методы с высокой оптимизацией выдают ясное и резкое изображение с практически незаметным размытием. Методы ресемплинга растровой графики написаны для огромного количества эмуляторов, 2D-игровых, к примеру, для AdvanceMAME, DOSBox. Все они имеют высокие оценки геймеров, которые портируют и переписывают игры, написанные в 80-х и 90-х. На данный момент (2015 г.) данные фильтры используются в сервисах XboxLive, VirtualConsole, и PSN для лучшего отображения старых игр на дисплеях с высоким разрешением. Примеры такихигр: Sonic's Ultimate Genesis Collection, Castlevania: The Dracula X Chronicles, Castlevania: Symphony of the Night, иAkumajō Dracula X Chi no Rondo'.

Что же мы имеем в итоге. Стало понятно, что развитие алгоритмов масштабирования пиксельной графики было возможно благодаря, в основном, использованию графических изображений в компьютерных играх. Раньше когда возможность вычислительных машин была небольшой, то довольствовались таким изображением, какое было, увеличивая изображение простыми алгоритмами. Когда же возможности стали возрастать, то захотелось сделать картинку более привлекательной для глаз. Обычных алгоритмов было недостаточно, пришлось придумывать новые, более эффективные.

В данной статье был проведен обзорный анализ алгоритмов масштабирования пиксельной графики. Каждый из них достаточно активно применялся в свое время, а некоторые используются и до сих пор. Сказать однозначно какой лучше – нельзя, так как в разных ситуациях ставятся различные требования к качеству масштабируемого изображения. Однако, можно с уверенностью констатировать, что рассмотренные новейшие алгоритмы имеют одно колоссальное отличие от классических алгоритмов масштабирования: они формируют сглаженную, плавную картинку, визуально убирая эффект ступенчатости.

На сегодняшний продолжается поиск оптимального алгоритма масштабирования. Главные критерии такого алгоритма можно назвать: качество формируемого изображения, минимум ресурсов вычислительной машины и скорость получения результата. Мощности компьютеров позволяют придумать сложные алгоритмы, осталось найти оптимальные методы.

Поиск алгоритма масштабирования входит в группу вопросов, которые сейчас вызывают наибольший интерес у специалистов. Авторы выражают надежду, что вскоре такой алгоритм будет найден, и проблема увеличения любой пиксельной картинки будет снята.

Список литературы

1. Алгоритмы масштабирования пиксельной графики // Статья википедии — свободной Интернет-энциклопедии — http://ru.wikipedia.org/wiki/ Алгоритмы_масштабирования
2. Абрамова О.Ф. Сравнительный анализ алгоритмов удаления невидимых линий и поверхностей, работающих в пространстве изображения [Электронный ресурс] / О.Ф. Абрамова, Н.С. Никонова // NovaInfo.Ru : электрон. журнал. - 2015. - № 38. – Режим доступа : http://novainfo.ru/archive/38/analiz-algoritmov-udaleniya-nevidimykh-liniy-i-poverkhnostey
3. Абрамова О.Ф., Котов В. В. К вопросу об импорте 3D моделей в программы с использованием графической библиотеки OpenGL [Электронный ресурс] / Котов В. В., Абрамова О.Ф. // Современная техника и технологии. - 2014. - № 1. - С. Режим доступа : http://technology.snauka.ru/2014/01/2965.
4. Александрина А.Ю. Разработка специализированных программных продуктов как форма научно-исследовательской работы студентов направления «Химическая технология» / А.Ю. Александрина, В.Ф. Каблов, О.Ф. Абрамова // Вестник Российского ун-та дружбы народов. Серия «Информатизация образования». - 2015. - № 4. - С. 59-66.
5. Трифанов А.И. Реализация собственного метода визуализации водной поверхности «скользящая текстура» / А.И. Трифанов, О.Ф. Абрамова // Современные наукоёмкие технологии. - 2013. - № 8 (ч. 1). - С. 96-97.
6. Шеховцов С. О., Исследование алгоритмов контентно-зависимого масштабирования изображений / Шеховцов С. О. - http://masters.donntu.org/2010/fknt/shekhovtsov/diss/index.htm
7. Thyssen A. Resize and Scaling. // Examples of ImageMagick Usage (Version 6), 2009. — http://www.imagemagick.org/Usage/resize/