УДК 681.327.12

А.И. Люлис

ВИЗУАЛИЗАЦИЯ КРУПНОФОРМАТНЫХ РАСТРОВЫХ ИЗОБРАЖЕНИЙ

Рассматриваются общие способы хранения, чтения и обработки растровых изображений. Отмечаются способы повышения производительности визуализации крупноформатных изображений. Предлагается способ визуализации крупноформатных растровых изображений с одновременной трансформацией в режиме реального времени для задач с динамично меняющимся интерфейсом.

Введение

Растровая форма представления изображений имеет широкое распространение среди пользователей цифровых технологий. Традиционно для их передачи, хранения и обработки используются бинарные файлы различных форматов [1, 2]. Большое разнообразие форматов файлов обусловлено различными областями их применения. Условно можно выделить две основные группы представления данных: в сжатой и несжатой формах. Сжатая форма данных в основном используется для обмена и хранения информации. Для ее обработки необходимо выполнить предварительное преобразование растра в массив, представленный в виде двухмерной матрицы, значения которой соответствуют яркостям пикселей растрового изображения. Необходимость выполнения таких преобразований объясняется простотой доступа к отдельным пикселям растрового изображения для повышения быстродействия обработки данных. В свою очередь, несжатая форма хранения данных использует уже готовые структуры, в которых положению каждого пикселя соответствует отдельная ячейка двухмерной матрицы. Несмотря на большие объемы, несжатая форма хранения данных широко используется в программных продуктах, где первоочередной задачей является повышение производительности обработки растровых изображений. К таким программным продуктам можно отнести всевозможные графические редакторы, геоинформационные комплексы и системы, векторизаторы и др.

Объектом исследования статьи являются данные, представленные в несжатой растровой форме. Их физические размеры составляют несколько десятков мегабайт. В основном данные таких объемов используются в геоинформационных системах для представления крупноформатных топографических карт с высоким разрешением. Разработка программных комплексов для их обработки требует реализации специальных алгоритмов, выполняющих эффективную обработку данных в режиме реального времени в условиях динамично меняющегося интерфейса.

1. Способы чтения растровых данных

Любая обработка растрового изображения начинается с его полной либо частичной загрузки в оперативную память компьютера. Считывание данных может быть выполнено с произвольного носителя информации (жесткого, оптического, электронного диска), загружено через Интернет либо получено непосредственно от устройства ввода (сканера, фотоприемной матрицы и др.). Изображения, полученные от устройств ввода, нередко имеют большие объемы и не всегда могут быть размещены непосредственно в оперативной памяти компьютера. Поэтому перед дальнейшей обработкой они сохраняются на жестком носителе. Далее будет рассматриваться только ситуация обработки данных с жестких носителей.

Время загрузки изображения или его части зависит от физического объема данных и от пропускной способности источника информации. Поэтому полная загрузка растра в основном используется для отображения небольших по размеру изображений (пиктограмм, фоновых рисунков) либо для непродолжительных манипуляций непосредственно с данными крупноформатного изображения.

Частичная загрузка растра используется для обработки крупноформатных изображений, размеры которых превышают максимальные размеры экрана в несколько раз. В этом случае в оперативную память загружается только часть изображения, необходимая для его отображения.

Такой способ обработки данных широко применяется в программах просмотра растровых изображений, в которых информация «подчитывается» из файла по мере надобности.

Основное место хранения растровых файлов – дисковые носители (винчестеры). Известно, что обращение к этим устройствам для выполнения операций чтения/записи требует существенных затрат машинного времени по сравнению с аналогичными операциями, которые выполняются непосредственно в оперативной памяти [3]. Поэтому для повышения производительности считывания данных предлагаются следующие рекомендации:

Снижение разрешения изображений. Как правило, это тривиальный способ увеличения производительности. Он заключается в уменьшении физических размеров изображения за счет снижения разрешения без ухудшения его качественных характеристик, пригодных для дальнейшей обработки.

Кодирование цвета. Обычно яркость пикселей цветного растрового изображения представлена прямым заданием интенсивностей составляющих r, g и b. Такой способ предпочтителен для изображений с плавными переходами цвета. В отдельных случаях яркость пикселей может быть представлена в 8-битовой псевдоцветной форме, где цвет пикселя преобразуется с использованием цветового куба, состоящего из 175 оттенков цвета (5 уровней красного, 7 уровней зеленого и 5 уровней синего) и 32 оттенков серого. Это позволяет уменьшить физические размеры изображения за счет снижения качества цветопередачи.

Сжатие данных [4, 5]. Этот способ позволяет сократить физический размер файла на жестком диске в несколько раз, что повышает скорость его считывания, но, как отмечено, для обработки потребуется дополнительное преобразование данных к нормальной несжатой форме.

Использование специальных форматов файлов. В основном обработка изображений осуществляется в оперативной памяти компьютера. Нередко полная загрузка крупноформатных растров непосредственно в оперативную память невозможна или неэффективна. В таких случаях данные обрабатываются небольшими частями путем дозагрузки с внешних носителей. Известно, что для эффективного выполнения дисковых операций немалое значение имеет последовательное считывание данных в оперативную память. Для этого используются специальные способы структурированного представления данных в файлах [6, 7].

Существует множество способов повышения производительности загрузки данных, но каждый из них в своей основе использует рассмотренные выше рекомендации, позволяющие оптимизировать работу с их источником. Так, в сети Интернет получил широкое распространение способ разделения исходного растра на множество изображений небольшого размера. В таком представлении каждый файл является частью большой растровой мозаики [8–10]. Список файлов мозаики с указанием их координатного расположения в единой системе координат хранится в специальном файле. Для обработки такого изображения достаточно выполнить загрузку необходимых фрагментов мозаики в соответствии с их положением в списке. Эта операция повышает производительность представления данных пользователю за счет загрузки в первую очередь видимой части изображения и дозагрузки оставшейся по мере необходимости.

Подобное хранение данных может быть представлено в виде отдельного файла, в котором каждому прямоугольному фрагменту изображения (блоку) соответствует определенное место в файле. Такая структура представления изображения в виде отдельных блоков внутри файла называется «черепичной» [6]. Каждый блок изображения имеет фиксированный размер последовательно размещенных данных. Положение отдельных блоков легко определяется исходя из их размеров. Для обработки требуемого фрагмента растра загружаются соответствующие блоки. Такая реализация позволяет оптимизировать обработку фрагментов изображения, но требует его разбивки на отдельные блоки для изображений, представленных в ином формате.

Другим способом повышения производительности обработки крупноформатных растров является построение пирамиды изображения [9, 11]. Способ первоначально был разработан применительно к задачам машинного зрения и сжатия изображений. Пирамида изображения представляет собой набор изображений в уменьшающемся масштабе, организованный в форме пирамиды (рис. 1). По мере движения вверх по пирамиде масштаб (размеры и разрешение) оригинала уменьшается. Целиком заполненная пирамида состоит из J+1 уровней, где $J = \log_2 N$, а $N \times N$ – размеры изображения. В большинстве случаев пирамида усекается до P+1 уровней, где

P лежит в пределах $1 \le P \le J$. Таким образом, исходное изображение в уменьшенном масштабе ограничивается P приближениями. Все уровни пирамиды могут быть сохранены в едином бинарном файле в виде набора приближений изображения с различными разрешениями.

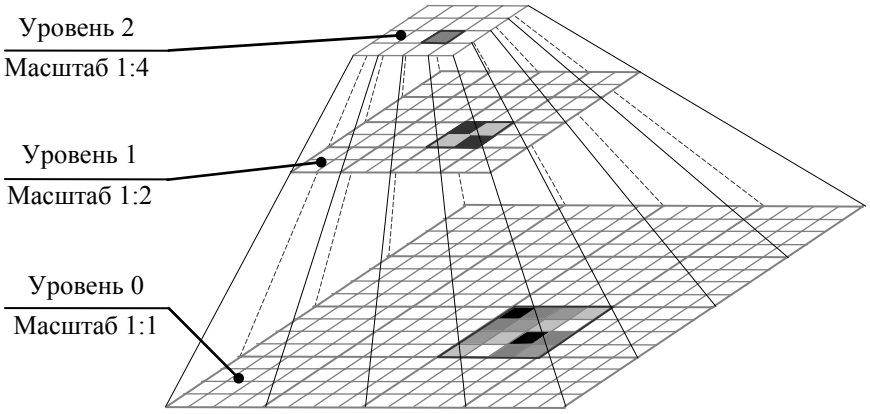

Рис.1. Пирамидальное представление растрового изображения

Из рисунка видно, что в основании пирамиды (уровень 0) лежит оригинал высокого разрешения, а вершина пирамиды состоит из приближений низкого разрешения (уровни 1 и 2). Такая организация построения данных позволяет в процессе обработки изображения осуществлять выборку приближений изображения с разрешением, близким к заданному уровню детализации (масштабу). Это повышает скорость чтения с дисковых носителей за счет минимизации обращений к ним и позволяет получить качественное отображение данных в произвольном масштабе. Избыточное чтение данных практически отсутствует, так как все операции выполняются с растром нужного приближения.

Нередко для обработки больших объемов данных операционной системой (например, Windows) предусмотрено прямое отображение файла в память. В таком случае нет необходимости в промежуточных операциях по размещению данных в памяти. Вместо этого выполняется их прямое отображение на адресное пространство приложения [3, 12], которое не требует от программиста принятия мер по оптимизации работы с носителем данных, так как все «берет на себя» механизм виртуальной памяти операционной системы. В результате происходит экономия ресурсов операционной системы и повышение скорости считывания данных.

2. Визуализация крупноформатных изображений

Функционирование большинства геоинформационных комплексов основано на обработке крупноформатных изображений. Большие размеры отдельных фрагментов объясняются высокими требованиями к их разрешению (300–600 dpi) для выполнения детальной обработки мелких объектов изображения. Основной решаемой задачей комплексов является оперативность получения результата обработки документов в режиме реального времени. Ниже рассмотрена оптимизация выполнения преобразований применительно к обработке крупноформатных изображений.

2.1. Масштабирование крупноформатного растра

Масштабирование растра является широко используемой операцией в обработке изображений [1]. Оно может применяться для отображения документа на экране монитора, когда его оригинальные размеры превышают максимальные размеры экрана, или для увеличения локальной области фрагмента изображения в процессе его детальной обработки. Из общеизвестных способов масштабирования можно выделить три основных [7]:

– *интерполяцию по ближайшему пикселю*. Отличается простотой реализации, но вместе с тем качество конечного результата не всегда удовлетворительно. Увеличение изображения приводит к возникновению ступенек, которые образуются за счет того, что непрерывные кривые изменения цвета становятся ступенчатыми, а уменьшение изображения приводит к исчезновению мелких деталей;

– *билинейную интерполяцию.* Выполняется на основе четырех соседних пикселей. Позволяет получить плавные края изображений, однако увеличенное изображение получается не резким;

– *бикубическую интерполяцию.* Осуществляется путем анализа 16 соседних пикселей, что обеспечивает более гладкое увеличение размеров. Дает несколько лучшие результаты, чем билинейная интерполяция.

Перечисленные способы приводят к дефектам в виде следов интерполяции. В последнее время появились более сложные алгоритмы (например, S-Spline), которые позволяют минимизировать эти дефекты. Они подвергают анализу бóльшую область вокруг каждого пикселя, чтобы качественно увеличить изображение, но такие решения требуют значительных временных затрат и используются при подготовке документа к печати путем его многократного увеличения.

Применение данных алгоритмов интерполяции дает удовлетворительный результат при увеличении изображения, но при его уменьшении в несколько раз эта операция эквивалентна прореживающей выборке с большим шагом. В таких случаях на глубоких уровнях масштабирования наблюдается ступенчатость контуров, так как выбранные пиксели иногда оказываются плохими представителями областей. Для получения качественного результата необходимо выполнить анализ бóльшего диапазона данных, представляющих результирующий пиксель, а это влечет за собой снижение производительности.

Для повышения производительности обработки изображения и сохранения его качества на низких уровнях предлагается использовать структуру представления изображения в виде пирамиды ее масштабных представлений. В этом случае для масштабирования изображения необходимо выбрать наиболее близкий уровень пирамиды и выполнить его обработку с применением ранее описанных способов интерполяции.

2.2. Линейные преобразования крупноформатных изображений

Кроме масштабирования (растяжения) нередко выполняются и другие линейные преобразования растра, например сдвиг, скос, вращение в плоскости, отражение относительно вертикальной либо горизонтальной осей. В обычных условиях преобразование растра изображения небольшого размера выполняется путем его отображения на подготовленный двухмерный растр с учетом коэффициентов аффинного преобразования.

В общем виде аффинное преобразование плоскости [13] в однородных координатах можно записать следующим образом:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} R_{11} & R_{12} & 0 \\ R_{21} & R_{22} & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix} + \begin{bmatrix} t_x & t_y & 1 \end{bmatrix}; \quad (1)$$

$$\begin{bmatrix} x' & y' \end{bmatrix} = R * \begin{bmatrix} x & y \end{bmatrix}, \quad (2)$$

где R_{ij} – элементы преобразования плоскости;

t_x, t_y – параметры переноса плоскости;

R – матрица преобразования плоскости;

x, y – координаты положения результирующего пикселя;

x', y' – координаты положения исходного пикселя.

Преобразование растра с учетом коэффициентов аффинного преобразования может быть выполнено с использованием кусочно-линейного алгоритма [14], который позволяет трансформировать произвольные выпуклые четырехугольники друг в друга. Так как каждое прямоугольное изображение может быть представлено парой его треугольных частей (треугольников), для каждого треугольника исходного изображения устанавливается треугольник, лежащий в плоскости результирующего изображения. Затем для каждой установленной пары треугольников вычисляются коэффициенты аффинного преобразования при переходе от одного треугольника к другому с помощью системы уравнений, полученной из (2):

$$\begin{cases} x' = R_{11}x + R_{12}y; \\ y' = R_{21}x + R_{22}y. \end{cases} \quad (3)$$

Для обработки изображения с учетом необходимых деформаций выполняется попиксельный обход подготовленного растра. Для каждого пикселя определяется треугольник, которому он принадлежит. На основании коэффициентов аффинного преобразования одного треугольника в другой (3) определяется значение яркости пикселя на исходном изображении, которое присваивается текущему пикселю.

При обработке крупноформатного изображения выделение большого объема данных для последующего заполнения яркостями пикселей исходного изображения неэффективно, что связано с высокими затратами системных ресурсов и низкой производительностью. Для решения этой задачи предлагается выполнять обработку растра постепенно, небольшими прямоугольными блоками. После заполнения каждого блока данными они могут быть сохранены в файл. Такой способ позволит выполнить обработку крупноформатного изображения без существенной нагрузки на системную часть вычислительной машины.

Как правило, описанная ситуация встречается при окончательной обработке данных. В реальных условиях результаты промежуточных преобразований данных должны быть представлены пользователю для дальнейшего анализа. В основном вывод данных осуществляется посредством окон операционной системы [3, 15]. Ограниченные размеры окон способствуют обработке крупноформатного изображения небольшими блоками, так как нет необходимости считывать те данные, которые не попадут в область видимости прямоугольного окна операционной системы.

3. Практическая реализация

Для визуализации крупноформатных растровых изображений предлагается использовать комбинированный способ, основанный на представлении изображения в виде пирамиды масштабных приближений с возможностью частичной обработки растра алгоритмом кусочно-линейного преобразования. Такая интеграция позволит повысить качество результата преобразования растра, уменьшенного в несколько раз, за счет обработки ближайшего к масштабу приближения растрового изображения, полученного из пирамиды. Для повышения быстродействия и рационального использования системных ресурсов обработка изображения выполняется небольшими прямоугольными блоками, в частных случаях не превышающими размеры экрана монитора.

В связи с тем что не каждый формат файла поддерживает механизм размещения пирамиды данных в едином файле, предлагается создать дополнительный файл, структура которого должна содержать представления изображения с различными разрешениями. При заполнении уровней пирамиды и вычислении яркостей пикселей предлагается использовать вид фильтрации с усреднением по окрестности [1, 7]. Это позволит получить качественный результат при уменьшении изображения в несколько раз. Программная реализация таких преобразований легко решается на уровне процессора в виде простейших логических и математических операций, что сокращает время создания пирамиды. Глубина представления яркости пикселей пирамиды может отличаться от глубины пикселей исходного изображения. Например, для изображений, представленных в градациях черного и белого цветов, для качественного отображения результата уровни пирамиды могут быть представлены в градациях серого цвета.

Для отображения данных исходного растра предлагается использовать 32-битное представление пикселя в формате RGBA независимо от исходного значения глубины цвета изображения. Данное предложение сделано из следующих соображений:

– представление исходного изображения в черно-белом формате позволит получить качественный результат при уменьшении оригинала в несколько раз;

– возможно сохранение дополнительной информации о прозрачности результата изображения для случаев, когда в процессе вычисления исходного положения пикселя результат окажется за границами изображения;

– 32-битное представление пикселя позволит ускорить копирование полученного результата преобразования в контекст устройства операционной системы [12], который часто использует такое же представление пикселя.

Для загрузки данных предлагается применять виртуальную память операционной системы с прямым отображением файла на адресное пространство приложения.

Кроме визуализации данных с линейным преобразованием растра может возникнуть необходимость его отображения с учетом произвольных нелинейных искажений. Для выполнения таких операций предлагается применять триангуляцию [16] изображения по заданному набору узловых точек. Деформация отдельных треугольных частей изображения на ограниченной прямоугольной поверхности не потребует больших временных затрат, но позволит получить качественный результат в любом масштабе.

Все рассмотренные рекомендации дадут возможность выполнить визуализацию крупноформатных изображений с высокой производительностью, качеством и минимальными системными требованиями. Алгоритм обработки данных предложенным способом может быть представлен в следующем виде:

1. Проверка существования файла с масштабными представлениями растра. Если таковой отсутствует, создать его.
2. Перечисление списка пар треугольников для преобразования изображения.
3. Определение области вывода результата для пары треугольников.
4. Вычисление коэффициентов аффинного преобразования для пары треугольников.
5. Отображение пикселей треугольной области с учетом коэффициентов преобразования.
6. Переход к п. 3, пока не будут отображены все треугольники, попадающие в прямоугольную область.

4. Оценки эффективности

Эффективность использования данного способа представлена в виде диаграмм, которые позволяют оценить загрузку процессора, эффективность использования буферов чтения-записи данных, физической памяти компьютера и файла подкачки операционной системы (рис. 2). Диаграммы получены с помощью программного обеспечения Process Explorer v11.33 [17]. Масштаб времени по горизонтали – 3 с в клетке.

На диаграммах представлено поочередное создание анонсов (100×100 пикселей) 13 изображений 2613×3692 пикселя и одного изображения 7083×7222 пикселя. Слева от разделителей (линии белого цвета) показано использование ресурсов компьютера с применением предложенного способа, справа – с применением средствами операционной системы Windows XP. Средние значения продолжительности выполнения данных операций представлены в таблице.

Pentium 4, 2.4 ГГц ОЗУ: 512 МБ	Размеры тестового изображения			
	2613×3692		7083×7222	
	Windows XP	Предложенный алгоритм	Windows XP	Предложенный алгоритм
Продолжительность формирования пирамиды, с	–	≈ 0,6	–	≈ 4
Продолжительность создания образа (100×100), с	≈ 2	≈ 0,01	≈ 6,5	≈ 0,01

Из диаграмм на рис. 2 следует, что производительность загрузки по предложенному способу существенно выше обычной, вместе с тем объем используемой физической памяти значительно меньше.

Обычно для повышения производительности обработки данных разработчики программного обеспечения используют прямое отображение файла в память. На диаграммах (рис. 3) представлено сравнение использования системных ресурсов при просмотре изображений предложенным способом и средствами программы ACDSee. Следует отметить, что во время просмотра данных выполнялось динамическое изменение масштаба. Возросшая нагрузка использования процессора видна на соответствующей диаграмме.

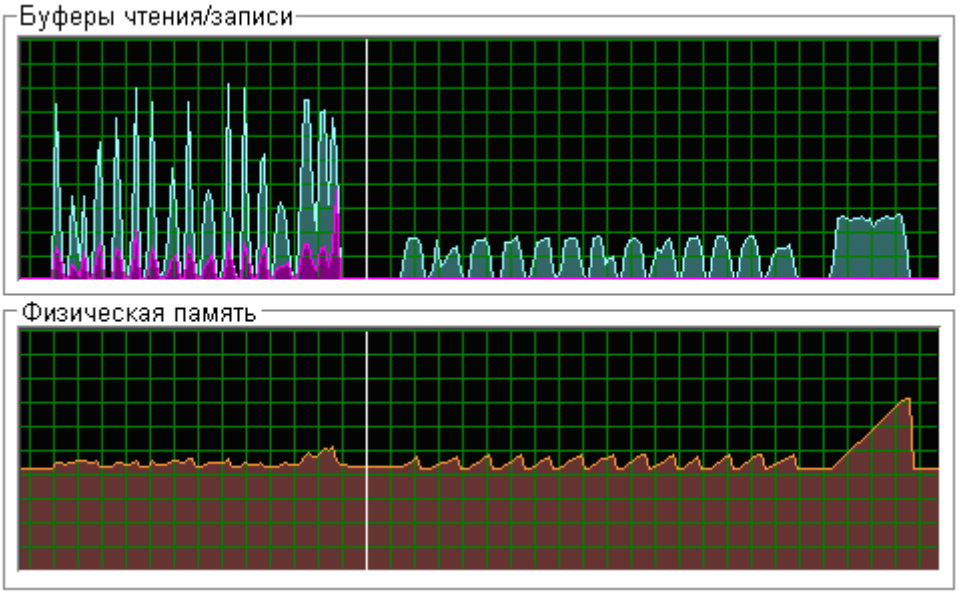

Рис. 2. Диаграммы использования буферов чтения-записи (3 МБ в клетке)
и физической памяти (50 МБ в клетке)

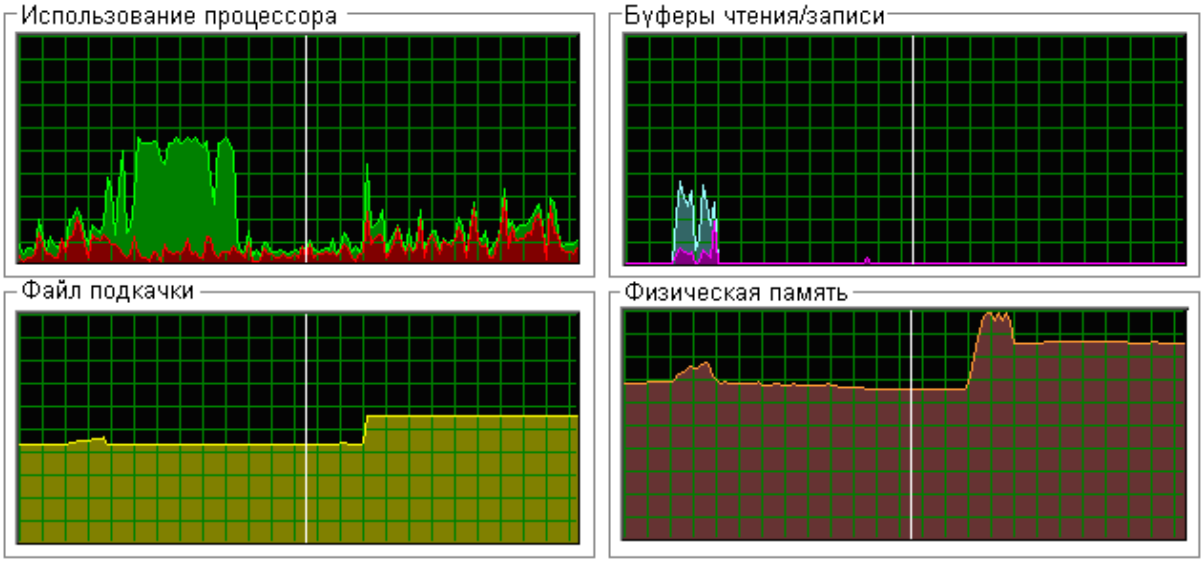

Рис. 3. Диаграммы работы предложенного способа и программного обеспечения ACDSee

На диаграммах слева от разделителей представлено использование системных ресурсов предложенным алгоритмом, справа – средствами ACDSee. Видно, что при практически одинаковых выполняемых операциях использование системных ресурсов (файла подкачки (100 МБ в клетке) и физической памяти (50 МБ в клетке)) средствами ACDSee практически соответствует размеру загружаемого файла. Такая реализация допустима при обработке одного изображения, но крайне неэффективна для нескольких изображений, если они в совокупном объеме превышают физическую память вычислительной машины.

Для оценки эффективности предложенного алгоритма при одновременной работе с несколькими крупноформатными растровыми изображениями представлены диаграммы поочередного формирования пирамиды четырех изображений 7083×7222 пикселя, 146 МБ (рис. 4), их визуализации в различных масштабах и вращение относительно опорной точки в режиме реального времени.

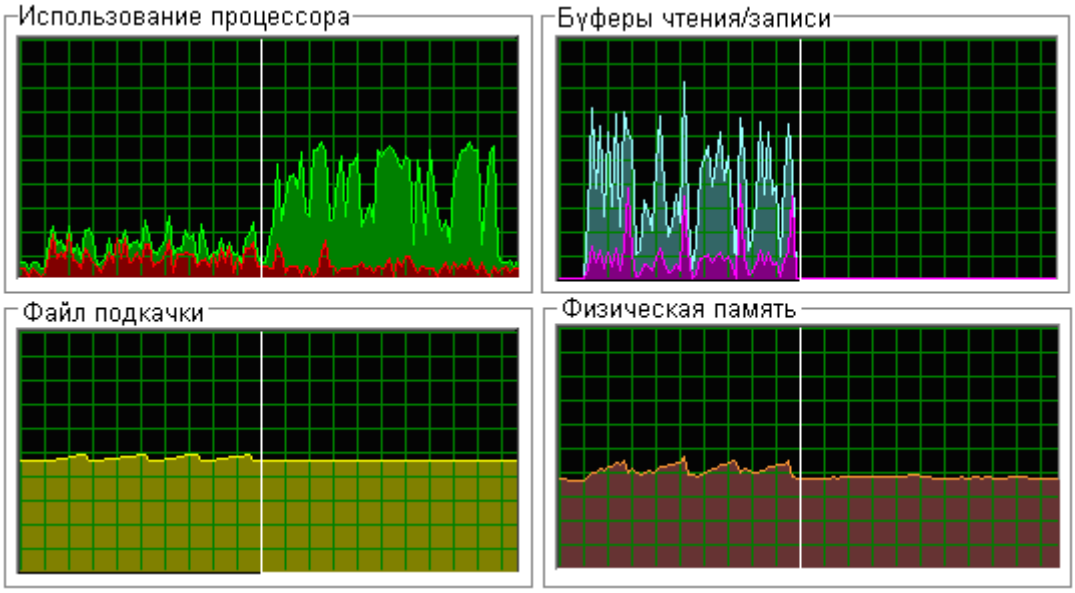

Рис. 4. Диаграммы использования системных ресурсов
при обработке крупноформатных растровых изображений

На диаграммах слева от разделителей показана загрузка четырех фрагментов, справа – выполнение аффинных преобразований. Видно, что выполнение дополнительных преобразований растровых изображений не сказывается на использовании дополнительных системных ресурсов по сравнению с рассмотренными способами. Это делает применение данного алгоритма эффективным в задачах, где необходима обработка данных, превышающих физические ресурсы вычислительных машин.

Заключение

В статье предложен способ обработки крупноформатных растровых изображений, основанный на представлении растра в виде пирамиды приближений изображения различного разрешения. Такое решение позволяет осуществить произвольное преобразование растра с ограниченным использованием системных ресурсов, высокой производительностью и качественным представлением результата в произвольном масштабе для решения задач динамического изменения интерфейса в режиме реального времени.

Способ реализован и опробован в аппаратно-программном комплексе семейства Дискан [18]. Данные комплексы предназначены для ввода крупноформатных документов малоформатным планшетным сканером по фрагментам. В отличие от существующих средств крупноформатного ввода (рулонные сканеры) с помощью данных комплексов можно ввести документы произвольных размеров с алюминиевых и диэлектрических (фанера, картон и др.) носителей. Это стало возможным благодаря технологии, в основе которой лежит визуализация процесса ввода документа на всех этапах – от задания произвольной области сканирования до окончательного формирования электронной копии документа. Для обеспечения конкурентоспособности и высокой производительности обработка данных осуществляется в режиме реального времени, что требует использования быстродействующих алгоритмов. Поиск общих точек в местах перекрытия произвольно расположенных фрагментов, деформация отдельных областей для получения качественного результата совмещения фрагментов, отображение схемы сканирования документа в произвольных масштабах – решение этих задач невозможно без использования специальных алгоритмов обработки крупноформатных растров. Предложенный способ позволил решить такие задачи с минимальными требованиями к системным ресурсам операционной системы. Это дало возможность использовать комплексы семейства Дискан практически на любых вычислительных машинах для получения качественной электронной копии документа.

Список литературы

1. Прэтт, У. Цифровая обработка изображений. Т. 2 / У. Прэтт. – М. : Мир, 1982. – 790 с.

2. Александров, В.В. Представление и обработка изображений: рекурсивный подход / В.В. Александров, Н.Д. Горский. – Л. : Наука, 1985. – 189 с.

3. Финогенов, К.Г. Win32. Основы программирования / К.Г. Финогенов. – 2-е изд. – М. : Диалог – МИФИ, 2006. – 416 с.

4. Климов, А.С. Форматы графических файлов / А.С. Климов. – Киев : НИПФ «ДиаСофт Лтд», 1995. – 480 с.

5. Wavelet transform for large scale image processing on modern microprocessors / D. Chaver [et al.] // VECPAR 2002. – 2003. – Vol. 2565. – P. 549–562.

6. Tiff Revision 6.0 Adobe Developers Association [Electronic resource]. – 1992. – Mode of access : http://partners.adobe.com/asn/developer/PDFS/TN/TIFF6.pdf. – Date of access : 10.03.2009.

7. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

8. Uytterhoeven, G. Wavelets: software and applications / G. Uytterhoeven [Electronic resource]. – Mode of access : http://users.telenet.be/geertu/PhD/. – Date of access : 13.03.2009.

9. Печников, А.О. Публикация растровых карт / А.О. Печников [Электронный ресурс]. – Режим доступа : http://www.citforum.ru/programming/digest/rastr_map/. – Дата доступа : 13.03.2009.

10. Jacob, J.C. Large-Scale Visualization of Digital Sky Surveys / J.C. Jacob, L.E. Husman // ASP Conference Proceedings «Virtual Observatories of the Future». – San Francisco : Astronomical Society of the Pacific, 2001. – Vol. 225. – P. 291–296.

11. Абламейко, С.В. Обработка изображений: технология, методы, применение / С.В. Абламейко, Д.М. Лагуновский. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1999. – 300 с.

12. Щупак, Ю.А. Win32 API. Эффективная разработка приложений / Ю.А. Щупак. – СПб. : Питер, 2007. – 572 с.

13. Шишкин, Е.В. Компьютерная графика / Е.В. Шишкин, А.В. Боресков. – М. : Диалог – МИФИ, 1995. – 288 с.

14. Роджерс, Д. Алгоритмические основы машинной графики / Д. Роджерс. – М. : Мир, 1989. – 502 с.

15. Ганаев, Р.М. Проектирование интерфейса пользователя средствами Win32 API / Р.М. Ганаев. – М. : Горячая Линия – Телеком, 2006. – 358 с.

16. Скворцов, А.В. Обзор алгоритмов построения триангуляции Делоне / А.В. Скворцов // Вычислительные методы и программирование. – 2002. – Т. 3. – С. 14–39.

17. Веб-узел усовершенствованных сервисных программ и технической информации [Электронный ресурс]. – Режим доступа : http://www.sysinternals.com. – Дата доступа : 13.03.2009.

18. Комбинированный ввод в компьютер крупноформатных изображений с использованием малоформатных планшетных сканеров и дигитайзеров / Г.И. Алексеев [и др.] // Информатика. – 2006. – № 4. – С. 71–78.

Поступила 18.05.09

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: lai@tut.by

A.I. Liulis

LARGESCALE IMAGE VISUALISATION

In this paper we consider the basic ways of storing, reading and processing the image data. The ways to improve the efficiency of image visualization are also considered. Large-scale image visualization with transformation in real time for the tasks with dynamically changing interface is proposed.