DOI: 10.15593/2499-9873/2018.4.05
УДК 519.714.2

С.С. Гусев

Институт проблем управления им. В.А. Трапезникова РАН,
Москва, Россия

ВЕКТОРИЗАЦИЯ РАСТРОВЫХ ИЗОБРАЖЕНИЙ

Статья посвящена разработке программного средства для векторизации цветных растровых изображений. Цель работы – создание приложения для перевода цветной растровой графики в векторный формат. Описывается алгоритм векторизации растровых изображений Potrace, приводятся описания векторных форматов Scalable Vector Graphics (SVG), Vector Markup Language (VML), Postscript, Adobe Portable Document Format (PDF), Adobe Illustrator.

Ключевые слова: векторная графика, растровая графика, векторизация, формат, алгоритм, программа.

S.S. Gusev

V.A. Trapeznikov Institute of Control Sciences
of the Russian Academy of Sciences,
Moscow, Russian Federation

VECTORIZATION OF RASTER IMAGES

This work is devoted to the development of program tools for vectorization of color bitmaps. The aim of the work is to develop an application for the translation of color raster graphics into vector format. The algorithm for vectorization of raster images Potrace is described, provides descriptions of vector formats Scalable Vector Graphics (SVG), Vector Markup Language (VML), Postscript, Adobe Portable Document Format (PDF), Adobe Illustrator.

Keywords: vector graphics, raster graphics, vectorization, format, algorithm, program.

Введение

Известно [1], что черно-белые изображения могут быть представлены как точечный или векторный рисунок. Точечный рисунок представляет изображение как сетку черных или белых пикселей. Векторное изображение можно описать через алгебраическое выражение его контуров в форме кривых Безье (Bezier). Преимущество векторного изображения – это возможность масштабирования до любого размера без потери качества. Векторные изображения независимы от разрешения любого специфического устройства. Они особенно популярны в очертании шрифтов, которым можно придать различный размер, на-

пример форматы векторных шрифтов, включающих шрифт PostScript Type 1, TrueType, и Metafont. Кроме того, это очень актуально в загружаемых и выгружаемых устройствах типа сканеров, мониторов, принтеров, в конечном счете устройствах, связанных с графикой.

Процесс преобразования векторного изображения в точечное называют *визуализацией (rendering)*. Обратный процесс превращения точечного изображения в векторное – *трассировка (tracing)*, или *векторизация*.

Ясно, что никакой алгоритм трассировки не совершенен в абсолютном смысле; так, может быть много всевозможных контуров, которые в конечном итоге могут привести к такому же самому точечному рисунку. Самые различные контуры могут быть применены к исходному точечному рисунку, они могут быть сходными с оригиналом или выглядеть более эстетично, нежели другие. Обычный способ визуализации точечных рисунков с высоким разрешением состоит в том, чтобы прорисовать каждый черный пиксель как точный квадрат, но это иногда приводит к контурным неровностям или так называемым «лестничным ступенькам». Ясно, что неровности [1] плохо воспринимаются, несмотря на приближенность изображения к оригиналу. Нет никакой абсолютной уверенности в том, что алгоритм трассировки будет оптимальным, но ясно, что именно такие алгоритмы дают лучшие результаты, чем подобные.

Среди алгоритмов векторизации выделяется алгоритм Potrace, который обладает большей точностью и быстродействием по сравнению с другими алгоритмами. Но его недостатком является отсутствие возможности векторизовать цветные растровые изображения. Поэтому возникла необходимость расширить функциональные возможности алгоритма путем разработки новой версии программного приложения.

1. Функциональные требования к компьютерной программе и решения по среде разработки

В 2002 г. автором была создана компьютерная программа, позволяющая векторизовывать цветные растровые изображении. Недостатками этой программы являлись: взаимодействие с векторизующим модулем через командную строку, поддержка только одного выходного формата, невозможность управления настройками алгоритма векторизации.

Новая версия программы должна давать пользователю возможность:

- открывать, просматривать и получать информацию об открытом растровом файле;
- уменьшать цветность, переводить в режим представления оттенками серого цвета, переводить в монохромный режим, отрытый растровый файл;
- сохранять открытый растровый файл с именем, отличным от исходного;
- настраивать параметры векторизации;
- векторизовывать открытый растровый файл;
- сохранять результат векторизации в выбранном пользователем векторном формате;
- открывать, просматривать и сохранять в выбранном формате созданный ранее векторный файл.

Для реализации этого функционала приложение должно:

- иметь средство отображения и сбора информации растрового файла;
- иметь средство изменения цветового режима растрового файла;
- иметь средство отображения SVG-файла;
- иметь средство конвертирования, записи и чтения векторных форматов.

Требования к интерфейсу приложения на этапе проектирования можно свести к следующему набору:

- приложение должно иметь пользовательский интерфейс, характерный для большинства приложений Windows, т.е. требуется наличие главного меню, панели инструментов;
- приложение должно обеспечивать визуальное отображение исходного (растрового) и результирующего (векторного) изображений.

В качестве среды разработки приложения было решено применить Borland Delphi 7. Для работы с растровыми файлами удобно использовать библиотеку Graphics32, которая позволяет удобно их просматривать и редактировать.

Для просмотра SVG-файлов на настоящий момент практически единственной альтернативой является Adobe SVGViewer 3.0. Данный ActiveX модуль можно включить в среду Delphi, что позволяет про-

сматривать векторные файлы формата SVG-файлы из разрабатываемого приложения.

Для поддержки дополнительных форматов решено использовать библиотеку PsToEdit, которая поддерживает конвертирование векторных файлов формата Postscript в ряд других форматов.

Для редактирования и компилирования библиотеки PsToEdit необходимо использовать среду разработки Microsoft Visual Studio .NET 2003.

В результате имеется следующий перечень ПО, используемого при разработке приложения:

- Delphi 7;
- Microsoft Visual Studio .NET 2003
- Adobe SVGViewer 3.0;
- PsToEdit.

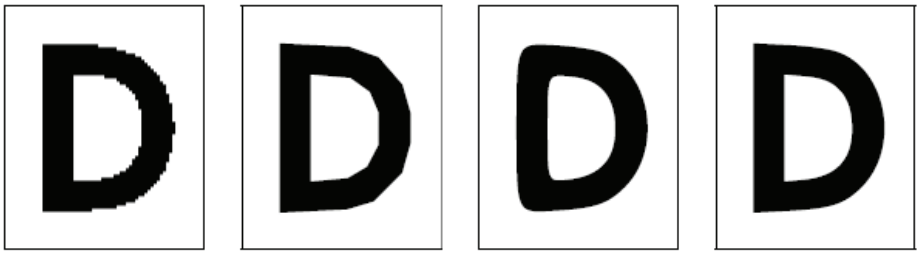
2. Алгоритм векторизации в приложении Potrace

Алгоритм Potrace создан для работы с изображениями высокого разрешения [2]. Сложные изображения, отсканированные с высоким разрешением, можно преобразовать в векторный контур, к примеру логотип компании или университетская эмблема. Другое возможное применение – это преобразование растровых шрифтов в контурные, если первоначальные растровые шрифты будут с высокой, достаточной разрешающей способностью. Никакой алгоритм трассировки не будет работать хорошо, если используется очень маленький масштаб изображения, рис. 1, *а–г*.

Это точечные рисунки для типичного 10pt экранного шрифта, представленного в разрешении 75 dpi (точек на дюйм). Однако этого достаточно для трассировки неточных форм, например для отсканированного, написанного от руки текста или рисунков мультипликации, если даже если они имеют относительно умеренные разрешающие способности.

Каждый оптимальный этап алгоритма трассировки должен выполнять несколько функций. Две из этих функций выполняют поиск самой вероятной кривой, которая приближает данные контуры к искомым обнаруженным углам. Существует связь между этими двумя функциями. Если обнаружено слишком много углов, результат трасси-

ровки будет напоминать негладкий многоугольник. Если также при обнаружении углов их будет немного, то контур рисунка будет выглядеть гладким, но округленным. Пример показан на рис. 1.



а *б* *в* *г*

Рис. 1. Обнаружение угла: *а* – оригинальный точечный рисунок; *б* – очень много углов; *в* – очень мало углов; *г* – оптимальное обнаружение угла

Другая важная функция, выполняемая алгоритмом трассировки, должна определить, какая особенность растрового изображения уместна и какие есть особенности артефактов (искажений) сканирования или процесса визуализации. Те особенности, которые можно считать артефактами, должны быть полностью отфильтрованы, поскольку если даже небольшое искажение остается, это может привести к видимым недостаткам по окончании трассировки. При этом прямая линия будет рассматриваться как выпяченная, очень маленькая и наклонная. Когда изображение представлено как точечный рисунок, такая линия будет в виде лестницы, где отдельные «шаги ступеньки» могут быть далеко друг от друга. Независимо от того, как далеко эти шаги отдалены друг от друга, на выходе должна быть прямая линия, иначе это вызовет визуальное искажение.

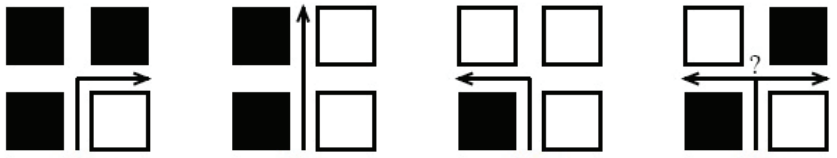
Алгоритм Potrace преобразовывает точечный рисунок в векторный контур за несколько этапов. Первоначальный этап – это расчленение точечного рисунка на несколько путей, которые формируют границы между черно-белыми областями. На втором этапе каждый путь приближен к оптимальному многоугольнику. На третьем этапе каждый многоугольник преобразован в сглаженную кривую. На четвертом (необязательном) этапе замыкающая кривая оптимизирована и соединяет вместе последовательные кривые Безье там, где это возможно. Наконец, результат сгенерирован в требуемый формат.

Рассмотрим каждый из означенных этапов более подробно.

2.1. Пути. Разбивка пути

Potrace использует следующий прямой метод анализа точечного рисунка в контуры. Сначала выбирается пара смежных пикселей различного цвета [3]. Это достигается, к примеру, выбором крайнего левого черного пикселя в каком-нибудь ряду. Соединение двух выбранных пикселей, формирующих край, осуществляется таким образом, чтобы черный пиксель был слева, а белый пиксель – справа. Этот край определяет длину пути.

Дальнейшее расширение этого пути происходит таким образом, что каждый новый край имеет черный пиксель слева и белый пиксель – справа относительно направления пути. Другими словами, при прохождении по граням между пикселями и каждый раз при достижении угла либо выбирается прямое направление, либо осуществляется поворот между левым или правым пикселем, в зависимости от цвета окружающих пикселей, как показано на рис. 2. Это происходит до тех пор, пока путь не дойдет до вершины, от которой он начинался, к точке, которая определяет закрытый путь.



а *б* *в* *г*

Рис. 2. Алгоритм разбивки пути

Каждый раз, когда определен закрытый путь, из графического контурного изображения удаляются и инвертируются все цвета пикселя в его внутренней области. Это определяет новый точечный рисунок, к которому применяется рекурсивный алгоритм к оставшимся там некоторым черным пикселям. В итоге образуются закрытые пути, которые будут переданы к следующей стадии алгоритма Potrace. В более поздних стадиях алгоритм Potrace рассматривает каждый путь самостоятельно.

2.2. Расчет поворота

В ситуации, описанной на рис. 2, *г*, указывается расчет того, произойдет ли левый поворот или правый поворот. Этот расчет не влияет на положительный или отрицательный результат алгоритма декомпозиции пути, поскольку любой путь будет замкнутым. Однако такой расчет оказывает влияние на форму закрытых выбранных путей.

В алгоритме Potrace варианты левого или правого поворота осуществляются *расчетом поворота*, который определяется через опцию командной строки, называемой *turnpolicy*.

Возможные расчеты поворота:

– *левый*, который всегда производит левый поворот;

– *правый*, который всегда производит правый поворот;

– *черный*, который предпочитает подключать черные компоненты пиксельного изображения;

– *белый* – белые компоненты;

– *меньшинство* – предпочитает подключить черный или белый цвет, являющийся наименее выраженным и находящийся в пределах данного соседства с текущей позицией;

– *большинство* – предпочитает подключить цвет, являющийся наиболее выраженным;

– *случайный* – производит случайный выбор цвета.

Заданный по умолчанию вариант поворота – *меньшинство*.

2.3. Оптимизация пути

Удаление пятен. Удаление пятен выполняется разбросом всех путей, внутренняя область которых имеет меньше, чем *t*, пикселей, для данного параметра *s*. Параметр *t* может быть представлен через опцию командной строки, называемую *turdsize*.

Многоугольники. Итогом работы данного этапа есть оптимальный многоугольник, имеющий сходство с очертаниями пути. В значении «оптимальный» подразумевается точный, а не приближенный многоугольник (рис. 3).

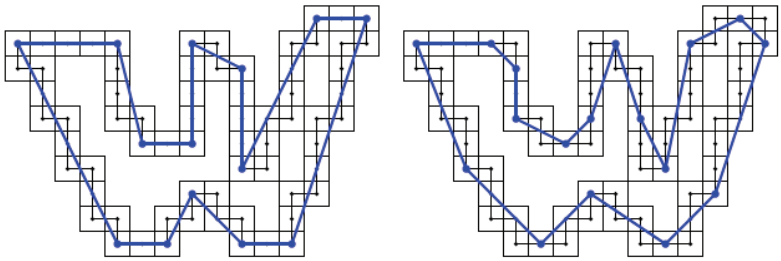


Рис. 3. Оптимизированный и неоптимизированный многоугольник пути

2.4. Получение векторного контура

Конечная стадия алгоритма – преобразование многоугольника, полученного в предыдущей стадии, в сглаженный векторный контур. На предварительном этапе происходила коррекция позиции вершины многоугольника для того, чтобы было наилучшее соответствие первоначальному точечному рисунку, насколько это возможно. На главном этапе определяются углы и кривые, основанные на длинах смежных сегментов линии и углах между ними.

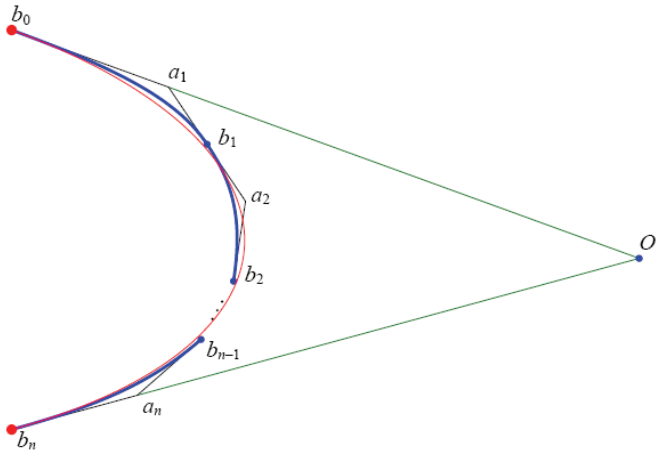


Рис. 4. Оптимизация кривой

Оптимизация кривой. Результатом предыдущей стадии алгоритма Potrace, после анализа угла и сглаживания, является кривая, состоящая из сегментов кривой Безье и сегментов прямой линии. Результирующая кривая очень близка к конечному результату алгоритма

Potrace. Однако есть еще одна дополнительная, последняя стадия алгоритма – стадия оптимизации кривой, которая делает попытку дальнейшего ее улучшения, соединяя смежные сегменты кривой Безье вместе там, где это возможно (рис. 4). Оптимизация кривой [4] делает незначительные изменения форм конечной кривой, и они настолько маленькие, что различий обычно не видно. Однако окончательная кривая состоит из меньшего количества сегментов и, таким образом, может быть представлена более сжато в финальной стадии программы. Если оптимизация кривой нежелательна, ее можно заблокировать запуском опции командной строки алгоритма Potrace, называемой *longcurve*.

2.5. Генерация результата

Масштабирование и вращение. Алгоритм Potrace производит семейство кривых, каждая из которых состоит из сегментов кривой Безье и сегментов прямой линии. Конечные и контрольные точки их сегментов являются произвольными точками в координатной плоскости [5]. В зависимости от выбранного ввода и параметров алгоритм Potrace выполнит линейное преобразование (масштабирование изображения к желательному размеру, возможно его вращать).

Кодирование избыточной информации. При использовании одного из PostScript алгоритма (PostScript или EPS) Potrace использует очень уплотненный числовой формат, чтобы преобразовать результат кривых Безье. Для этого требуется много параметров кривой. В принципе необходимо 6 параметров для описания каждого сегмента кривой Безье (одна конечная точка и две контрольных точки). Однако если параметров мало, Potrace может кодировать каждый сегмент, используя только 3–4 реальных числа.

Кодирование избыточности параметров кривых Безье всегда выполняется со встроенным алгоритмом PostScript, который использует макровозможности языка PostScript. Также кодирование избыточности параметров может быть выключено опцией командной строки *longcoding*, оно происходит дольше по времени, но имеет более приемлемый результат.

Квантование. Для большинства вычислений конечные координаты, которые являются реальными числами, квантуются и округляются до 1/10 пикселя. Таким образом, номер десятичных цифр необходи-

мо представлять как каждую уменьшенную координату для эффективного размещения всех контрольных точек на сетке координат. Координаты точек можно тогда представлять как целые числа.

Заданная по умолчанию константа квантования 1/10 обычно дает хорошие результаты, однако это происходит с перестраиванием конфигурации через опцию командной строки *unit*.

2.6. Пример окончательного результата алгоритма программы

Пример окончательного выполнения алгоритма Potrace показан на рис. 5.

Рис. 5, *а* показывает первоначальное точечное изображение. На рис. 5, *б* обратим внимание, как заданное по умолчанию «меньшинство» просчитывает поворот, сохраняя черные контуры по внешней стороне замкнутой фигуры, в то же время сохраняет белые области в месте прорисовки волос на рисунке. Отметим, что на области волос было удалено пятнышко размером с единицу. Рис. 5, *б* также показывает оптимальный многоугольник, рассчитанный для каждого компонента пути.

Рис. 5, *в* показывает откорректированные вершины многоугольника, относительно оригинального точечного рисунка, который показан серым цветом. Каждая вершина окружена ее квадратным модулем. На этом этапе происходит анализ углов для данного точечного рисунка, но обратим внимание, что очень немногие углы обнаружены. Вообще анализ угла происходит лучше при более высоком разрешении точечного рисунка. На рисунке синим цветом показаны сглаженные кривые, замыкающие сегменты кривой Безье и сегменты линий.

На рис. 5, *г* представлен результат оптимизации кривой: первоначальная кривая показана синим цветом, оптимизированная кривая – красным цветом. Красные точки являются индикаторами новых границ сегмента, при этом количество сегментов уменьшилось со 112 до 68, т.е. на 40 %. Конечный результат алгоритма показан на рис. 5, *д*.

Отладка результата на рис. 5, *б*, *в* и *г* может быть произведена за счет различных вариантов командной строки, используемых в Potrace, – от d1 до d3.

Рис. 5. Пример окончательного результата алгоритма программы: *а* – первоначальный точечный рисунок; *б* – декомпозиция пути и оптимальный многоугольник; *в* – настройка вершины, анализ угла, и сглаживание; *г* – оптимизация кривых; *д* – конечный результат

3. Векторные форматы

Формат SVG. Формат SVG (Scalable Vector Graphics – масштабируемая векторная графика) – это новый XML-словарь, предназначенный для описания двухмерной векторной графики для Интернета и других приложений.

SVG был разработан Консорциумом W3C. Консорциум W3C – некоммерческая организация, занимающаяся разработкой открытых

стандартов, например HTML и XML, а также многих других важных словарей. В создании SVG участвовало более 20 организаций, включая Sun Microsystems, Adobe, Apple, IBM и Kodak.

Компания Sun с самого начала принимала участие в разработке спецификации SVG, и в рабочей группе экспертов имеется два активных ее представителя.

Поскольку SVG является форматом XML, SVG-графика может легко генерироваться веб-серверами «на лету» с использованием стандартных средств XML, многие из которых написаны на Java. Например, веб-сервер может генерировать высококачественные и при этом небольшие по объему графики на основе оперативных биржевых сводок.

SVG дает возможность создавать графику с помощью специальных графических пакетов или автоматически (например, используя JavaServer Pages). SVG позволяет легко манипулировать графикой с помощью стандартных инструментов XML.

Возможность работы в различных средах – ключевое преимущество SVG. Например, с помощью генератора двухмерной графики SVG любое Java-приложение может экспортировать графику в формате SVG. И затем эти изображения могут импортироваться, просматриваться и модифицироваться в различных средах.

Формат Postscript. Postscript был разработан Джоном Уорноком и Чаком Гешке из Adobe Systems в начале 80-х гг. Исходно Postscript использовался как ядро механизма печати компьютеров Apple, но вскоре стал широко распространенным стандартом для большинства компьютерных систем. Интерпретаторы Postscript (в виде программных или аппаратных компонентов) для печати документов присутствуют практически во всех современных компьютерных системах. В Postscript используется модель изображения текста (или рисунков) на чистой странице. Когда страница готова, она выводится на печать, и начинается «прорисовка» изображения очередной страницы. Это есть не что иное, как метод компиляции. Каждый документ Postscript включает в себя программу, которая печатает на принтере (или отображает на экране монитора) следующие друг за другом страницы.

PostScript соединил в себе лучшие возможности принтеров и плоттеров. Подобно плоттерам, PostScript предоставляет возможность вывода высококачественной векторной графики и единый язык управления, который может быть использован любым производителем

принтеров. Подобно матричным принтерам, PostScript предлагает удобные возможности по печати растровой графики и текста. В отличие от тех и других, PostScript может совмещать все эти типы вывода на одной странице, давая намного больше гибкости, чем до этого имел любой принтер или плоттер.

PostScript – больше, чем типичный язык управления принтером, он является полнофункциональным языком программирования. Многие прикладные программы могут преобразовать документ в PostScript-программу, при выполнении которой будет получен начальный документ. Эта программа может быть послана непосредственно на принтер с поддержкой PostScript или преобразована интерпретатором PostScript в другой формат (для принтеров без поддержки PostScript), либо результат ее выполнения интепретатором может быть показан на экране. Поскольку исходная PostScript-программа одна и та же, PostScript называется *независимым от устройства*.

Большинство высокопроизводительных принтеров и плоттеров имеют встроенный интерпретатор языка PostScript. В то же время простые принтеры домашнего класса поддерживают только элементарные графические операции, поэтому задача создания растрового изображения возлагается на центральный процессор. Существуют интерпретаторы языка PostScript для различных операционных систем, наиболее известный из них – свободная программа Ghostscript.

Формат VML. VML (Vector Markup Language) – это XML-приложение, разработанное корпорацией Microsoft для кодирования двумерной векторной графики. Поддержка VML в настоящее время осуществляется только обозревателем Internet Explorer 5.x и пакетом Microsoft Office 2000.

Microsoft передал в мае 1998 г. спецификацию этого языка W3C в качестве предложения для стандартизации, однако W3C предложил в качестве стандарта собственную спецификацию SVG (Scalable Vector Graphics). Полной программной реализации SVG пока нет, но этот язык, несомненно, будет в ближайшее время поддержан многими системами.

Приведенное ниже описание VML соответствует его описанию в MSDN Library. Мы описываем только те возможности VML, которые поддерживаются веб-обозревателем; расширения VML, добавленные в Microsoft Office 2000, не рассматриваются. Для того чтобы видеть,

как элементы VML отображаются на экране, эту и последующие главы следует просматривать в обозревателе Internet Explorer 5.x.

Формат PDF. PDF (аббревиатура от англ. Portable Document Format) – кроссплатформенный формат электронных документов, созданный фирмой Adobe Systems с использованием ряда возможностей языка PostScript. В первую очередь предназначен для представления в электронном виде полиграфической продукции – значительное количество современного профессионального печатного оборудования может обрабатывать PDF непосредственно. Для просмотра можно использовать официальную бесплатную программу Acrobat Reader, а также программы сторонних разработчиков. Для создания PDF-документов требуется собственническая shareware-программа Adobe Systems – Adobe Acrobat либо программы сторонних разработчиков.

Траектории в PDF описывают контуры объектов. Цвет, которым будет заполнен контур, указывается оператором „R G B rg“, где R, G и B – интенсивность красного, зеленого и синего цветов соответственно (0 – минимальная интенсивность, 1 – максимальная). Далее следует список операторов, по одному на строку. Вначале этого списка следует оператор „m“, который определяет координаты начальной точки траектории (формат – „X Y m“). Затем – оператор „l“, который проводит линию до определенной точки (формат – „X Y l“), или оператор „c“, рисующий кривую Безье (формат – „X1 Y1 X2 Y2 X3 Y3“ c, первые две указанные точки – опорные, третья – конечная точка, начальной точкой является крайнее положение предыдущей фигуры). Замыкание контура осуществляются командой „h“, заливка – командой „f“.

Формат Adobe Illustrator. Adobe Illustrator, первоначально разработанный для платформы Macintosh, – известная и широко используемая программа создания изображений. Существуют версии для Macintosh, Microsoft Windows и NeXT. Мощные возможности Adobe Illustrator обусловлены тем, что в качестве графических объектов здесь реализованы кривые Безье, а также наличием простого пользовательского интерфейса, который обеспечивает точное позиционирование сплайновых графических объектов. Использование кривых Безье дает некоторые преимущества при моделировании естественных (а в определенных случаях и искусственных) объектов, файлы Adobe Illustrator применяются для обмена графическими элементами.

Траектории в данном формате описывают контуры объектов. Цвет, которым будет заполнен контур, указывается оператором „C M Y K k“, где C, M, Y и K – интенсивность цветов цветовой модели CMYK. Начало описания очередного контура обозначается оператором „*u“, далее следует список операторов, по одному на строку. Вначале этого списка следует оператор „m“, который определяет координаты начальной точки траектории (формат – „X Y m“). Затем – оператор „l“, который проводит линию до определенной точки (формат – „X Y l“) или оператор „c“, рисующий кривую Безье (формат – „X1 Y1 X2 Y2 X3 Y3 curveto“, первые две указанные точки – опорные, третья – конечная точка, начальной точкой является крайнее положение предыдущей фигуры). Замыкание контура и его заливка осуществляются командой „f“.

Заключение

Рассмотрен принцип векторизации растровых изображений, основные векторные форматы.

Разработана новая версия программы, векторизующей цветные растровые изображения, в которой были исправлены недостатки первой версии. Векторизующий модуль был встроен в программу, что позволило увеличить скорость получения результата более чем на порядок.

Реализована возможность сохранять результат в одном из нескольких векторных форматов, открывать сохраненные ранее результаты, уменьшать количество цветов исходного изображения, также добавлена поддержка 32-битных изображений.

Список литературы

1. Иванова Н.Ю., Малинин А.А., Таяновская Ю.Б. Метод инвариантного анализа изображений, заданных в векторной форме // Научно-технический вестник информационных технологий, механики и оптики. – 2006. – С. 188–192.

2. Горошкин А.Н. Применение векторного подхода к распознаванию рукописных символов // Вестник Сибирского государственного аэрокосмического университета имени академика М.Ф. Решетнева. – 2006. – С. 15–17.

3. Пролубников А.В. Интервальный подход к решению задачи распознавания числовых матриц // Вычислительные технологии. – 2012. – Т. 17, № 4. – С. 77–87.

4. Молчанова В.С. Адаптивный пороговый метод бинаризации растровых изображений технических чертежей // Прогрессивные информационные технологии. – 2015. – С. 62 – 70.

5. Терехин А.В. Распознавание объектов методом вычисления оценок с использованием диагональных признаков формы // Технические науки. Информатика, вычислительная техника. – 2014. – № 1(29). – С. 17–25.

References

1. Ivanova N.YU., Malinin A.A., Tayanovskaya YU.B. Metod invariantnogo analiza izobrazhenij, zadannyh v vektornoj forme [The method of invariant analysis of images given in vector form]. Nauchno-tekhnicheskij vestnik informacionnyh tekhnologij, mekhaniki i optiki. 2006. 188 – 192 pp.

2. Goroshkin A.N. Primenenie vektornogo podhoda k raspoznavaniyu rukopisnyh simvolov [Apply a vector approach to recognition of handwritten characters]. Vestnik Sibirskogo gosudarstvennogo aehrokosmicheskogo universiteta imeni akademika M.F. Reshetneva. 2006. 15 – 17 pp.

3. Prolubnikov A.V. Interval'nyj podhod k resheniyu zadachi raspoznavaniya chislovyh matric [Interval approach to solving the problem of recognition of numerical matrices]. Vychislitel'nye tekhnologii. 2012. T. 17. iss. 4, 77 – 87 pp.

4. Molchanova V.S. Adaptivnyj porogovyj metod binarizacii rastrovyh izobrazhenij tekhnicheskih chertezhej [Adaptive threshold method for binarization of the raster images of technical drawings]. Progressivnye informacionnye tekhnologii. 2015. 62 – 70 pp.

5. Terekhin A.V. Raspoznavanie ob"ektov metodom vychisleniya ocenok s is-pol'zovaniem diagonal'nyh priznakov formy [Object recognition by calculating estimates using the diagonal shape features]. Tekhnicheskie nauki. Informatika, vychislitel'naya tekhnika. iss. 1(29). 2014. 17 – 25 pp.

Получено 03.06.2018

Об авторе

Гусев Сергей Сергеевич (Москва, Россия) – соискатель Института проблем управления им. В.А. Трапезникова РАН (117997, г. Москва, ул. Профсоюзная, 65, e-mail: gs-serg@mail.ru).

About the author

Sergei S. Gusev (Moscow, Russian Federation) – Ph.D. Student, V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences (65, Profsouznaya st., Moscow, 117997, Russian Federation, e-mail: gs-serg@mail.ru).