УДК 681.007.05

АЛГОРИТМ ВЕКТОРИЗАЦИИ РАСТРОВЫХ ИЗОБРАЖЕНИЙ МЕТОДОМ ПОИСКА ГРАФИЧЕСКИХ ПРИМИТИВОВ

А.А. Журов, к.т.н. А.В. Липанов
(представил д.т.н., проф. А.В. Королёв)

Рассматривается алгоритм построения векторной модели полноцветного растрового изображения. Векторная модель предполагает получение изображения в векторном виде, максимально близкого к своему растровому оригиналу. Векторный вид изображения предполагает, что элементы изображения будут описаны набором линий, кривых, а также некоторым числом точек, которые не удастся отнести к линиям. Данный алгоритм реализован в программе, при помощи которой осуществляется исследование разработанного алгоритма.

Введение. Задача построения векторной модели растрового изображения весьма **актуальна**, поскольку в различных приложениях требуется получать именно векторное представление изображения. Примером такой задачи может быть векторизация отсканированного чертежа для дальнейшего использования в CAD-системах. Преимущество хранения графической информации в векторном виде – изображение в векторном виде требует, как правило, меньшего места на носителях информации. **Основной проблемой** при преобразовании из растрового формата в векторный является достижение 100% сходства между исходным и конечным изображением. Существует достаточно большое количество алгоритмов векторизации полутоновых изображений [1 – 5], но для векторизации полноцветных изображений алгоритмы практически не описаны. Предлагаемый алгоритм рассчитан на решение задачи векторизации полноцветного растрового изображения. Алгоритм преобразует изображение в набор графических примитивов трех видов: точки, прямые и кривые Безье. Прямые также делятся на три вида: горизонтальные, вертикальные и наклонные. Рассмотрим процесс получения указанных выше графических примитивов, которые при объединении дают исходное изображение, хотя и с некоторыми погрешностями. Алгоритмы поиска графических примитивов используют результаты, описанные в [1].

1	2	3
8		4
7	6	5

Рис. 1. Точка и ее соседи (8-связные компоненты)

Точка. Под точкой понимается пиксель, не имеющий соседей одинакового цвета. Под соседними пикселями понимаются пиксели, окружающие исходный (рис. 1).

Пусть B – множество пикселей изображения, B(i,j) – некоторая точка изображения, для которой осуществляется формирование множества P точек изображения и множества L_h горизонтальных линий. Каждый элемент множества P представляет собой точку с координатами x, y и определенным цветом. Формирование множества P осуществляется следующим образом:

$$p_i = B(i,j) \mid \left\{ \begin{array}{l} B(i,j) <> B(i+1,j), B(i,j) <> B(i-1,j); \\ B(i,j) <> B(i,j-1), B(i,j) <> B(i,j+1); \\ B(i,j) <> B(i-1,j-1), B(i,j) <> B(i+1,j-1); \\ B(i,j) <> B(i-1,j+1), B(i,j) <> B(i+1,j+1) \end{array} \right\}, P = \bigcup_i p_i, \quad (1)$$

где p_i – точка во множестве P.

Далее для наглядности будем использовать цветом фона – белый, а цветом изображения – черный.

Горизонтальная прямая. Под горизонтальной прямой понимается совокупность соседних пикселей одинакового цвета, имеющих общую координату x (рис. 2). Во множестве L_h каждый элемент – это линия, представленная координатами начала и конца, а также цветом. Формирование множества L_h осуществляется следующим образом:

$$l_i = l_i \cup B(i,j) \mid B(i+1,j) = B(i,j), \; L_h = \bigcup_i l_i, \quad (2)$$

Рис. 2. Горизонтальная прямая

где l_i – линия во множестве L_h.

Вертикальная прямая. Под вертикальной прямой понимается совокупность соседних пикселей одинакового цвета, имеющих одинаковую координату Y (рис. 3). Множество вертикальных прямых обозначим L_v. В данном случае элементом множества L_v является вертикальная линия. Следует отметить, что при поиске вертикальных линий анализу подвергаются только те точки изображения, которые не были замаркированы, т.е. занесены во множество M при построении горизонтальных линий. Формирование данного множества осуществляется следующим образом:

$$l_i = l_i \cup B(i,j) \mid \{B(i,j+1) = B(i,j), B(i,j) \notin M\}, \; L_v = \bigcup_i l_i \; . \quad (3)$$

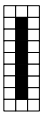


Рис. 3. Вертикальная прямая

Наклонная прямая. Под наклонной прямой подразумевается совокупность соседних пикселей одного цвета, образующих совместно линию в каком-либо направлении (рис. 4). На рис. 4 используется направление 5, указанное на рис. 1.

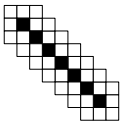


Рис. 4. Наклонная прямая

Следует отметить, что описываемый алгоритм позволяет осуществлять поиск линий с углом наклона ± 45° и требует дальнейшего обобще-

ния для поиска линий с различным углом наклона. Формирование множества наклонных линий L_k осуществляется следующим образом:

$$l_i = l_i \bigcup B(i,j) \mid \left\{ \begin{matrix} (B(i+1,j+1) = B(i,j)) \bigcap (B(i-1,j+1) = B(i,j)), \\ B(i,j) \notin M \end{matrix} \right\}, \quad L_k = \bigcup_i l_i. \quad (4)$$

При осуществлении процесса формирования линий по формулам (2) – (4) на каждом шаге после добавления новой точки к линии изменяются координаты конца линии, т.е. фактически координаты новой добавленной на i-м шаге точки становятся координатами конца линии. Координаты начала линии – это координаты первой точки, которая была отнесена к некоторой линии.

Кривая Безье. Для обеспечения эффективного процесса векторизации сложных кривых линий используется метод векторизации с использованием кривых Безье. В качестве кривой Безье понимается совокупность прямых, образующих непрерывную кривую, причем количество подряд идущих пикселей с одинаковой координатой x либо y не должно превышать 5, такие же условия накладываются на наклонные прямые, образующие кривые Безье. Пример кривой Безье показан на рис. 5.

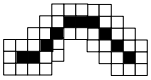


Рис. 5. Последовательность точек, которые будут отнесены к кривой Безье

Как известно, кривые Безье [2 – 4] являются очень популярными в области обработки изображений. Кратко напомним, что представляют собой кривые Безье. Рассмотрим совокупность N + 1 опорных точек P_k (k = 0, N) в трехмерном пространстве. Параметрическая функция кривой Безье имеет вид

$$B(U) = \sum_{k=0}^{N} p_k \left[\frac{N!}{k!(N-k)!} u^k (1-u)^{N-k} \right], \qquad u \in [0,1]. \qquad (5)$$

Выражение в квадратных скобках в (5) – функция стыковки тогда, когда она смешивает контрольные точки, чтобы сформировать кривую Безье. B(u) – непрерывная функция в трехмерном пространстве, которая описывает кривую с N дискретными опорными точками P_k. В первой опорной точке u = 0 (k = 0), во всех последующих опорных точках u = 1 (k = N).

Известны основные правила для построения кривых Безье.

1. Кривая не проходит ни через одну из опорных точек за исключением 1-й и последней. Для вышеуказанной формулы: $B(0) = P_0$ и $B(1) = P_N$.

2. Кривая всегда содержится в пределах выпуклого множества контрольных точек; она никогда сильно не отклоняется от контрольных точек.

3. Если имеется только одна опорная точка P_0, т. е. N = 0, тогда $B(u) = P_0$ для всех u.

4. Если имеются только две опорные точки P_0 и P_1, т. е. N = 1, то формула преобразуется в линию между этими двумя точками.

5. Невозможно построить кривую Безье, параллельную другой кривой Безье. Кривой Безье невозможно описать окружность и невозможно создать кривую Безье, параллельную другой, за исключением случаев совпадения параллельных кривых или кривых прямой линии Безье.

Функция стыковки – это многочлен, порядок которого ниже, чем число контрольных точек. Поэтому три контрольных точки приводят к параболе, а четыре контрольные точки – к кубической кривой, и т.п. Слияние первой опорной точки и последней образует замкнутую кривую. Одна непрерывность порядка может достигаться, если тангенсы между первыми двумя точками и последними двумя точками равны. При многократном добавлении опорных точек в одно и то же место, происходит увеличение его «веса» и кривая Безье начинает «тянуться» к этому месту.

Под увеличением числа контрольных точек понимаются многочлены высшего порядка и возможно более высокие факториалы. Это необходимо для того, чтобы можно было соединить маленькие участки кривых Безье и сформировать более длинную кривую. Это также поможет контролировать местные состояния. Зачастую, изменение месторасположения одной контрольной точки может привести к деформации всей кривой. Кривая начинается и заканчивается в начальной и конечной контрольных точках, что позволяет физически легко соответствовать секциям. Возможно также соответствие первым производным, начиная с тангенсов углов наклона линий, соединяющих начальную и конечную контрольные точки. Существуют исключительные случаи, однако для описываемого алгоритма они значения не имеют.

Также существуют и другие методы, в работе которых заложены методы построения кривых Безье по меньшему количеству опорных точек.

Построим кривую Безье, описанную тремя опорными точками: $P_{20}(x_{20}, y_{20})$, $P_{21}(x_{21}, y_{21})$ и $P_{22}(x_{22}, y_{22})$. Она должна быть представленна кривой Безье третьего порядка (кубической кривой Безье). Существуют следующие опорные точки для кривой третьего порядка: $P_{30}(x_{30}, y_{30})$, $P_{31}(x_{31}, y_{31})$, $P_{32}(x_{32}, y_{32})$, $P_{33}(x_{33}, y_{33})$. Опорные точки должны иметь следующий вид: $x_{30} = x_{20}$ и $y_{30} = y_{20}$, $x_{33} = x_{22}$ и $y_{33} = y_{22}$. Так как квадратичная кривая Безье имеет три опорные точки, мы можем использовать за основу либо функцию Бернштейна, либо функцию B-spline. Функция Бернштейна эквивалентна B-spline при условии, если N = M, где N + 1 – число опорных точек и M – это порядок кривой Безье. То же самое применяется к кубической кривой, которая имеет четыре опорные точки. Так, квадратичная кривая $P_2(t)$, где $0 \leq t \leq 1$, может быть записана как:

$$x_2(t) = x_{20}\, B_{02}(t) + x_{21}\, B_{12}(t) + x_{22}\, B_{22}(t);$$
$$y_2(t) = y_{20}\, B_{02}(t) + y_{21}\, B_{12}(t) + y_{22}\, B_{22}(t), \qquad (6)$$

где $B_{kn}(t)$ является основной функцией Бернштейна, которая имеет вид

$$B_{kn}(t) = \frac{n! x^k (1-x)^{n-k}}{k!(n-k)!}.$$

Подобно описывается кубическая кривая $P_3(t)$, где $0 \le t \le 1$:

$$x_3(t) = x_{30}\, B_{03}(t) + x_{31}\, B_{13}(t) + x_{32}\, B_{23}(t) + x_{33}\, B_{33}(t);$$
$$y_3(t) = y_{30}\, B_{03}(t) + y_{31}\, B_{13}(t) + y_{32}\, B_{23}(t) + y_{33}\, B_{33}(t). \qquad (7)$$

Из приведенного выше описания кривых Безье следует, что для построения кривой Безье третьего порядка необходимо иметь две опорные точки и две направляющие. Исходя из этого, необходимо строить наборы точек, которые будут представлять собой опорные и направляющие точки для некоторой последовательности точек изображения. При построении кривой Безье она должна представлять собой исходный набор точек изображения, использованных для ее построения. При построении кривой Безье используются выражения (7).

Для определения некоторой точки, которая может быть опорной для некоторой кривой Безье G_i, используется следующая формула:

$$g_0 = B(i, j) \mid \{(B(i+1, j+1] = B(j, i)) \cap (B(i-1, j+1] = B[i, j])\}. \qquad (8)$$

Для определения промежуточных точек, которые могут быть отнесены к данной кривой Безье и которые могут быть использованы как направляющие и опорные точки, используется следующее правило:

После прохода по всему изображению уже имеется информация о всех точках и прямых. Производится анализ на смежность прямых и точек. Если есть смежные, то они объединяются в кривые и перемаркировываются уже как кривые Безье. Запоминается точка начала – это будет первая опорная точка (точка A, рис. 6). Последняя опорная точка является концевой точкой последней прямой (точка D, рис. 6). Во время объединения прямых в кривую происходит анализ направления изгиба кривой. Первая направляющая точка определяется как 1/4 расстояния от начальной опорной точки по горизонтали и 1/5 расстояния от самой высокой точки кривой. Если направляющая точка находится ниже кривой, то 1/5 расстояния делится пополам и проверяется, выше или ниже кривой новая координата точки, если выше, то координата остается, если нет – то про-

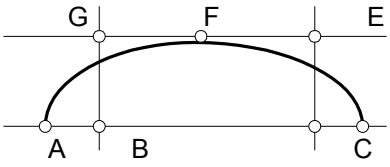


Рис. 6. Процесс поиска точек для кривой Безье

цесс повторяется, пока не будет достигнут необходимый результат. Вторая направляющая точка находится идентично первой, только за начальную точку берется вторая опорная.

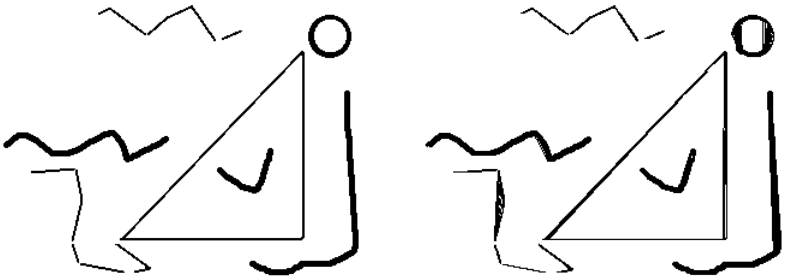
Таким образом, формируется кривая Безье G_i, которая затем добавляется в множество кривых Безье данного изображения. В результате обработки изображения формируются множества точек, прямых и кривых Безье, которые в совокупности и представляют собой исходное изображение, но уже в векторном виде.

Для проверки работоспособности алгоритма разработана программа, которая позволяет строить векторную модель растрового изображения. Результат работы программы показан на рис. 7.

Как видно из рис. 7, полученная векторная модель и исходное изображение практически идентичны. Для оценки меры сходства изображений используется следующий метод вычисления погрешности:

$$k = \frac{100 * (W * H - \Delta)}{W * H}, \text{ где } \Delta = \sum_{j=}^{H} \sum_{i=1}^{W} \left| B_1(i,j) - B_2(i,j) \right|, \qquad (9)$$

k – погрешность векторизации; W – ширина изображения; H – высота изображения; Δ – коэффициент корреляции изображений; $B_1(i,j)$ – исходное изображение; $B_2(i,j)$ – растровый аналог векторизованного изображения.



а б

Рис. 7. Результат работы программы:
а – исходное изображение, б – векторная модель

Из (9) видно, что при величине коэффициента корреляции [5], равном нулю, k равно 100%, что означает полную идентичность исходного и векторизованного изображений. При выборе других коэффициентов корреляции [5] выражение для вычисления k также должно быть изменено. Для изображений, приведенных на рис. 7, погрешность векторизации k = 98,5.

Блок-схема алгоритма векторизации приведена на рис. 8.

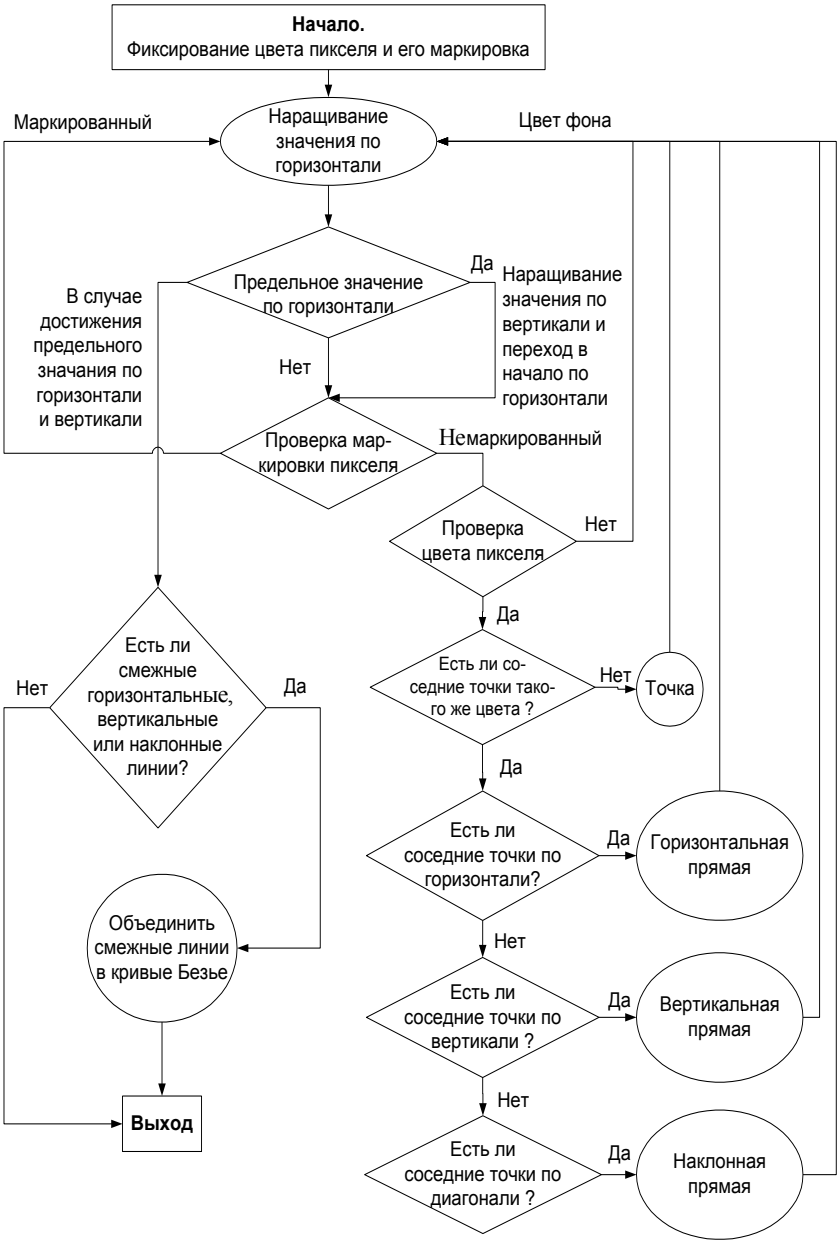


Рис. 8. Блок-схема алгоритма

Тестирование скорости работы алгоритма показало, что он имеет достаточно высокую скорость обработки изображения, для вышеуказанных изображений векторизация происходила 0,410 секунды для конфигурации компьютера: процессор Intel Celeron 700 Mhz, 256 Mb ОЗУ.

Описанный алгоритм векторизации целесообразно использовать для изображений «скелетного» типа, то есть не содержащих обширные области заливки. Также не рекомендуется векторизовать изображения с градиентными заливками. Это связано с тем, что данный алгоритм не обладает необходимой устойчивостью при поиске областей с заливками, в том числе и градиентными.

В качестве перспективных путей развития данного алгоритма следует указать необходимость совершенствования пути отбора точек, принадлежащих различным графическим примитивам, а также добавление в алгоритм возможностей поддержки прямоугольников, как графических примитивов. Также в дальнейшем предполагается доработка алгоритма для расширения спектра обрабатываемых изображений, в первую очередь повышения сложности обрабатываемых изображений, исследование алгоритма на устойчивость к случайным помехам и добавление возможностей фильтрации помех.

ЛИТЕРАТУРА

1. *Липанов А.В., Путятин Е.П. Интегрально-разностный алгоритм выделения контуров для систем технического зрения // Системи обробки інформації. – Х.: НАНУ, ПАНМ, ХВУ. – 2000. – Вип. 3(9). – С. 129 – 133.*
2. *Роджерс Д. Алгоритмические основы машинной графики. Пер. с англ. – М.: Мир, 1989. – 512 с.*
3. *Прэтт У. Цифровая обработка изображений: Пер. с англ. в 2-х книгах. – М.: Мир, 1982.– 736 с.*
4. *Вельтмандер П.В. Основные алгоритмы компьютерной графики. Учебное пособие в 3-х книгах. Книга 2. Машинная графика. – Н-ск: Новосибирский государственный университет, 1997. – 382 с.*
5. *Липанов А.В., Путятин Е.П. Исследование алгоритмов обнаружения объектов на основе методов корреляционного распознавания и алгоритма параллельной нормализации // Радиоэлектроника и информатика. – Х.: ХТУРЭ. – 1998. – № 3. – С. 118 – 122.*

Поступила 9.04.2004

***ЖУРОВ Андрей Андреевич**, факультет ПММ Харьковского национального университета радиоэлектроники.*

***ЛИПАНОВ Александр Витальевич**, канд. техн. наук, факультет ПММ Харьковского национального университета радиоэлектроники.*