УДК 681.3.06

Ю.Л. Костюк, Ю.Э. Абдуллин, А.А. Чертов

ПЕРВИЧНАЯ ВЕКТОРИЗАЦИЯ МНОГОЦВЕТНЫХ РАСТРОВ С ИСПОЛЬЗОВАНИЕМ ТРИАНГУЛЯЦИИ И ПРОЦЕДУРЫ ПОСТОБРАБОТКИ

Развивается метод векторизации с использованием триангуляции. Метод распространяется на изображения, состоящие из линий, точек и областей, закрашенных как постоянными, так и непрерывно изменяющимися цветами. Кроме того, рассматриваются алгоритмы постобработки, повышающие качество распознавания линий.

Векторизация с использованием триангуляции была предложена в работах [1, 2]. В них рассматриваются растровые изображения, состоящие из сравнительно небольшого числа цветов. При этом смежные пиксели, принадлежащие одной и той же области, должны быть окрашены одним цветом. В [2] описан алгоритм построения граничных линий (ломаных), разделяющих все изображение на замкнутые области, каждая из которых окрашена своим цветом. Далее на звеньях этих ломаных строится триангуляция с ограничениями, и внутри треугольников строятся центральные (скелетные) линии. По скелетным линиям и границам оценивается толщина исходных (распознаваемых) линий, а также принимается решение, что та или иная область изображения является площадным объектом. Однако при вычислении скелетных линий могут получаться нежелательные эффекты: паразитные боковые ответвления, излишние осцилляции и др.

В настоящей статье рассматривается случай более сложных изображений, в которых цвет пикселей одной области может изменяться в некоторых пределах. Кроме того, рассматриваются процедуры постобработки скелетных линий для борьбы с нежелательными эффектами.

Векторизация растра с непрерывно изменяющимися цветами

Цвет пикселя задается тройкой неотрицательных чисел – его цветовыми координатами $\{r, g, b\}$. Пусть цветовое расстояние ρ_{ij} между парами пикселей $\{r_i, g_i, b_i\}$ и $\{r_j, g_j, b_j\}$ задано некоторой подходящей функцией, например $\rho_{ij} = \max(|r_i - r_j|, |g_i - g_j|, |b_i - b_j|)$, или в виде евклидова расстояния.

Будем считать, что два соседних пикселя (по горизонтали, вертикали или диагонали) принадлежат одной и той же области с непрерывно изменяющимся цветом, если цветовое расстояние между ними меньше заданного порога $e1$. Тогда небольшая модификация алгоритма [2] на шаге 1 позволит по растру вычислять набор граничных ступенчатых линий между областями с существенно различающимися цветами. На рис. 1 показан пример растра с двумя цветами пикселей. Следует отметить, что в отличие от изображений с областями строго одинакового цвета в этом случае некоторые граничные линии могут оказаться незамкнутыми.

На шаге 2 выполняется генерализация (аппроксимация) граничных ступенчатых линий аналогично алгоритму [2] (рис. 2). Результатом являются отрезки ломаных, при этом дополнительно для каждого отрезка фиксируется ориентация и запоминаются цвета четырех пикселей: слева в начале отрезка, справа в начале, слева в конце и справа в конце. Для того чтобы в дальнейшем по цветам этих пикселей можно было точнее восстановить цвет внутри каждой области, нужно при запоминании производить усреднение по нескольким соседним пикселям.

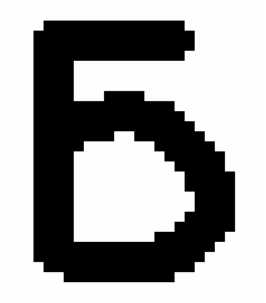

Рис. 1. Векторизуемый бинарный растр. Граничные ступенчатые линии разделяют области с отличающимися цветами

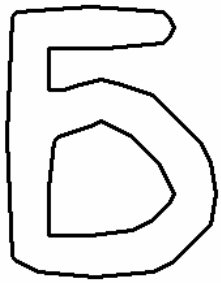

Рис. 2. Аппроксимация граничных ступенчатых линий

На шаге 3 строится триангуляция с ограничениями, в которой полученные на предыдущем шаге отрезки ломаных используются как структурные линии (рис. 3). Известные алгоритмы [3] решают эту задачу за гарантированное время $O(n \log n)$ и за $O(n)$ в среднем, где n – количество отрезков. Полученные при этом дополнительные отрезки будем называть невидимыми.

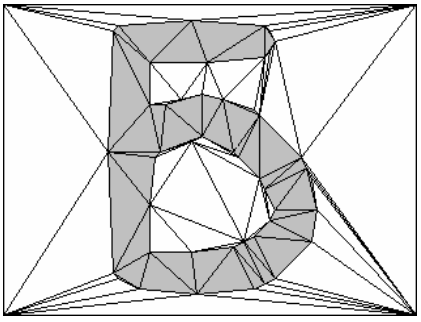

Рис. 3. Триангуляция с ограничениями, построенная по отрезкам

На шаге 4 для каждого треугольника строятся скелетные линии. Если у треугольника одно невидимое ребро, то в скелет добавляется отрезок, соединяющий середину невидимого ребра и противолежащую вершину. Если у черного треугольника два невидимых ребра, то добавляется отрезок, соединяющий их середины. Если невидимых рёбер три, то добавляются отрезки, соединяющие центр масс треугольника с серединой каждого из невидимых рёбер (рис. 4).

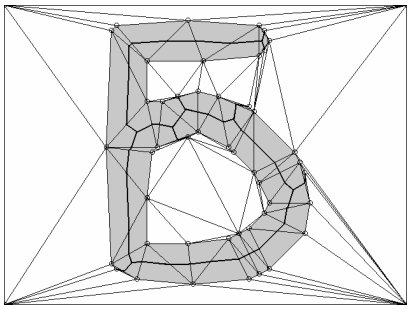

Рис. 4. Скелетные линии, построенные по триангуляции (показаны линии внутри только лишь одной области)

На шаге 5 для каждого треугольника оценивается толщина распознаваемой линии. Если через треугольник проходит одно ребро скелета, то этому ребру присваивается толщина, равная отношению площади треугольника к длине этого ребра. Если же рёбер скелета три, то каждому из них присваивается толщина $S/3L_i$, где S – площадь треугольника, а L_i – длина ребра (рис. 5).

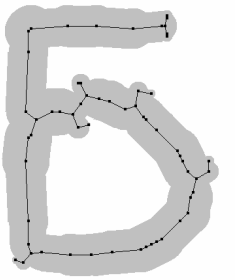

Рис. 5. Скелетные линии после вычисления ширины

На шаге 6 производится классификация треугольников на следующие группы:

1) треугольники, которые представляют точечный объект на исходном изображении;

2) треугольники, принадлежащие некоторому площадному объекту;

3) треугольники, принадлежащие некоторой линии на исходном изображении.

Классификация производится на основе оценки толщины и длины скелетной линии для цепочки смежных треугольников. Если толщина больше заданного параметра $t1$, то этот треугольник принадлежит площадному объекту, если толщина меньше параметра $t1$, а длина не превышает заданный параметр $t2$, то это – точечный объект, иначе это – линия на исходном изображении.

На этом исполнение основного алгоритма заканчивается. Дальнейшая обработка зависит от того, к какому классу отнесена каждая из выделенных областей, состоящая из группы смежных треугольников.

Для области – точечного объекта обработка состоит в оценке координат центра, вычислении среднего диаметра и вычислении усредненного цвета.

Для области – площадного объекта обработка состоит в формировании границ области, запоминании номеров треугольников, цветов пикселей в вершинах треугольников.

Далее будем рассматривать обработку треугольников, распознанных как линия на исходном изображении. Основные проблемы, которые при этом встречаются, – это ложные ответвления, сильные колебания толщины линии и слишком большое число вершин скелетных линий, из-за которых осциллирует вся линия.

Алгоритм устранения ложных ответвлений

Алгоритм удаления ложных ответвлений является фильтром: в зависимости от параметров ответвления он либо допускает его, либо делает пометку, что ответвление должно быть устранено. В качестве параметров проще всего использовать длину и среднюю толщину ответвления. Сам алгоритм приведён ниже.

Шаг 1. Выбирается очередная непросмотренная вершина скелета степени 1 (вершина v). Если все такие вершины просмотрены, переход на шаг 5.

Шаг 2. От вершины v в цикле делается переход к следующей вершине до тех пор, пока не встретится вершина со степенью, не равной 2 (либо «развилка», либо другая концевая вершина). Обозначим конечную вершину w.

Шаг 3. Если степень вершины w равна 1, то делается переход на шаг 1.

Шаг 4. Если степень вершины w равна 3 и более, то необходимо принять решение, удалять данное ответвление или нет. В качестве критерия можно, например, взять отношение средней толщины ответвления к его длине. Если эта величина больше некоторого порогового значения, то ответвление помечается на удаление, иначе – нет. Переход на шаг 1.

Шаг 5. Удаляются все помеченные ответвления.

Конец алгоритма.

Данный алгоритм работает за время, пропорциональное числу вершин. Скелетные линии после удаления ответвлений изображены на рис. 6.

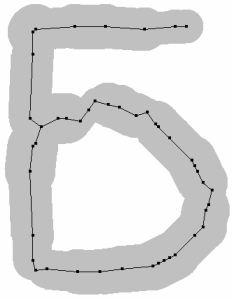

Рис. 6. Удаление ответвлений по критерию «средняя ширина/длина > 0,3»

На практике важно сделать процедуру выбора значения управляющего параметра как можно более удобной для пользователя. Желательно, чтобы можно было оценить (например, по небольшому фрагменту изображения), какой результат получится, если выбрать то или иное число.

Алгоритм сглаживания толщины линии

Для сглаживания толщины линии предлагается простейший «жадный» алгоритм. В процессе его работы обработанными будут называться те рёбра, которые вошли в какую-либо цепочку рёбер, для которой усреднялась толщина.

Алгоритм состоит из следующих шагов.

Шаг 1. Выбирается какая-либо вершина v_0, с которой смежно нечётное число необработанных ребёр (по аналогии с алгоритмом нахождения эйлерова пути). Если таких вершин нет, то v_0 – произвольная вершина, с которой смежно необработанное ребро. Выберем любое необработанное ребро, смежное с v_0. Вершину, смежную с v_0 по этому ребру, обозначим v_1.

Шаг 2. При помощи жадного алгоритма цепочка $v_0, v_1, \ldots, v_n$ достраивается до тех пор, пока отношение минимальной и максимальной толщины её рёбер не превышает некоторого значения, например 2. В очередной точке переход осуществляется по первому подходящему необработанному ребру.

Шаг 3. Средняя толщина для данной цепочки вычисляется как $w=(w_1l_1+w_2l_2+\ldots+w_nl_n)/(l_1+l_2+\ldots+l_n)$, где w_i, l_i – ширина и длина i-го ребра цепочки. Все ребра цепочки помечаются как обработанные, ширина каждого ребра устанавливается равной w.

Шаг 4. Если остались необработанные ребра, то переходим к шагу 1.

Конец алгоритма.

Пример работы алгоритма представлен на рис. 7.

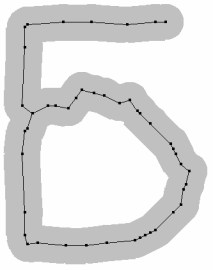

Рис. 7. Результат работы алгоритма сглаживания толщины

Для скелетов, построенных при помощи триангуляции, описанный алгоритм работает за линейное время.

Алгоритм генерализации, основанный на методе наименьших квадратов

Чтобы уменьшить число вершин в скелете, необходимо применить некоторый алгоритм генерализации. Алгоритм, который будет приведён ниже, основан на методе наименьших квадратов (МНК).

Пусть на плоскости задано множество точек $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$. Также задана некоторая прямая l, на которой лежит точка $\vec{x}$ и направление которой задаётся вектором $\vec{a}$ единичной длины. Сумма квадратов расстояний от точек до прямой может быть записана как $J(\vec{x},\vec{a}) = \sum \left|\vec{a}\times(\vec{x}-\vec{x}_i)\right|^2$, где «×» – векторное произведение. Поиск аппроксимирующей прямой по МНК состоит в минимизации J по $\vec{x}$ и $\vec{a}$. Данная задача решается известными методами [4] за время O(N). Если зафиксировать $\vec{x}$ и проводить оптимизацию только по $\vec{a}$, то решение задачи не усложняется.

В целом алгоритм сглаживания ломаной будет следующим.

Шаг 1. На вход подаётся ломаная $P_1, P_2, \ldots, P_n$. Индексу текущей вершины i присваивается 1.

Шаг 2. Пока $i \neq n$, выполнять следующие действия:

2.1. Находится максимальное j, такое что аппроксимирующая прямая, полученная по МНК для множества точек $\{P_i, P_{i+1}, \ldots, P_j\}$, отстоит от каждой из этих точек на расстояние, не большее ε (управляющий параметр). Если i=1, то требуется, чтобы прямая проходила через P_1, аналогично, если $j=n$, то требуется, чтобы прямая проходила через P_n.

2.2. Найденная прямая «запоминается», переменной i присваивается значение j.

Шаг 3. Построение результирующей ломаной. В цикле просматриваются все пары последовательных аппроксимирующих прямых:

3.1. Пусть первая из прямых l_1 аппроксимирует множество точек $\{P_i, P_{i+1}, \ldots, P_j\}$, а следующая прямая l_2 – множество $\{P_j, P_{j+1}, \ldots, P_k\}$. Выясняется, имеют ли эти две прямые точку пересечения.

3.2. Если имеется точка пересечения и она отстоит от P_j не более, чем на ε, то она добавляется в результирующую ломаную, иначе добавляются две точки: перпендикуляр из P_j на l_1 и перпендикуляр из P_{j+1} на l_2.

Шаг 4. В результирующую ломаную добавляются начальная точка P_1 и конечная точка P_n.

Конец алгоритма.

При сглаживании скелет разбивается на множество ломаных – замкнутых и незамкнутых. На замкнутых ломаных искусственно задаётся начальная вершина, которая и является концевой. После этого к каждой из ломаных необходимо применить алгоритм генерализации (рис. 8).

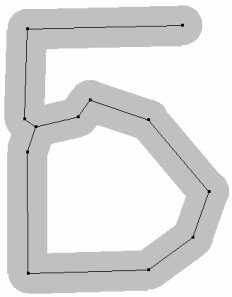

Рис. 8. Результат сглаживания скелетной линии (ε = 8)

В отличие от многих известных алгоритмов генерализации [5, 6], точки результирующей ломаной могут не лежать на исходной ломаной. В этом смысле приведенный алгоритм является более общим.

Трудоёмкость алгоритма генерализации составляет $O(n^2)$. Если предположить, что при добавлении новых точек во множество, по которому делается аппроксимация, максимальное отклонение от найденной прямой монотонно растёт, то на шаге 2.1 вместо линейного поиска можно использовать бинарный. Это снизит трудоёмкость до линейно-логарифмической за счёт незначительного увеличения числа звеньев в генерализованнной ломаной.

Описанные алгоритмы постобработки распознанных линий могут существенно повысить качество векторизации. Они были реализованы в программе-векторизаторе и проверены на ряде примеров.

ЛИТЕРАТУРА

1. *Костюк Ю.Л., Новиков Ю.Л.* Графовые модели цветных растровых изображений высокого разрешения // Вестник Томского государственного университета. 2002. № 275. С. 153–160.
2. *Костюк Ю.Л., Кон А.Б., Новиков Ю.Л.* Алгоритмы векторизации цветных растровых изображений на основе триангуляции и их реализация // Вестник Томского государственного университета. 2003. № 280. С. 275–280.
3. *Скворцов А.В., Костюк Ю.Л.* Применение триангуляции для решения задач вычислительной геометрии // Геоинформатика. Теория и практика. Вып. 1. Томск: Изд-во Том. ун-та, 1998. С. 127–138.
4. *Weisstein E.* Least Squares Fitting-Perpendicular Offsets [Электронный ресурс]: MathWorld, a Wolfram Web Resource. – Режим доступа: http://mathworld.wolfram.com/LeastSquaresFittingPerpendicularOffsets.html, свободный.
5. *Jimenez J., Navalon J.* Some Experiments in Image Vectorization [Электронный ресурс]: IBM Journal of Research and Development. – Режим доступа: http://www.research.ibm.com/journal/rd/266/ibmrd2606J.pdf, свободный.
6. *Chen D., Daescu O.* Space-Efficient Algorithms for Approximating Polygonal Curves in Two-Dimensional Space [Электронный ресурс]: поисковая база данных CiteSeer. – Режим доступа: http://citeseer.ist.psu.edu/chen98spaceefficient.html, свободный.

Статья представлена кафедрой теоретических основ информатики факультета информатики Томского государственного университета, поступила в научную редакцию «Информатика» 15 июня 2006 г.