ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 13–26

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.021

ПОСТРОЕНИЕ ВНУТРЕННЕЙ ДИАГРАММЫ ВОРОНОГО МНОГОУГОЛЬНОЙ ФИГУРЫ МЕТОДОМ ЗАМЕТАНИЯ

© 2024 г. Л. М. Местецкий[a, b, *], Д. А. Коптелов[a, **]

[a] *Московский государственный университет им. М. В. Ломоносова*
119991 Москва, ГСП-1, Ленинские горы, д. 1, Россия
[b] *Национальный исследовательский университет "Высшая школа экономики"*
109028 Москва, Покровский бульвар, д. 11, Россия
**E-mail: mestlm@mail.ru*
***E-mail: dimitar98@list.ru*

Поступила в редакцию: 01.09.2023
После доработки: 15.03.2024
Принята к публикации: 15.03.2024

В статье рассматривается задача построения внутренней диаграммы Вороного многоугольной фигуры – многоугольника с многоугольными дырами. Предлагается метод, основанный на парадигме плоского заметания. Прямое построение только внутренней части диаграммы Вороного позволяет уменьшить объем вычислений и повысить робастность по сравнению с известными решениями. Другим фактором снижения вычислительной сложности является использование свойства попарной инцидентности линейных отрезков, образованных сторонами многоугольной фигуры. Для учета этих особенностей предлагается строить структуру данных статус заметающей прямой в виде упорядоченного множества зон ответственности сайтов. Структура реализуется в виде комбинации сбалансированного дерева и двунаправленного списка. Вычислительные эксперименты иллюстрируют численную надежность и эффективность предложенного метода.

Ключевые слова: диаграмма Вороного

DOI: 10.31857/S0132347424040026, **EDN:** PTJOCY

1. ВВЕДЕНИЕ

Многоугольная фигура (МФ) представляет собой замкнутую область на плоскости, граница которой состоит из непересекающихся простых многоугольников, т. е. это многоугольник с многоугольными дырами. Скелетом или срединной осью такой фигуры является множество точек центров максимальных кругов, содержащихся внутри фигуры. Скелеты находят применение в задачах распознавания формы, в частности, для распознавания рукописного текста по цифровым изображениям. Растровое изображение текста аппроксимируется многоугольными фигурами, и для этого множества многосвязных фигур строится скелет (рис. 1) [1]. В примере на рис. 1 аппроксимация дает 4 фигуры, их границы состоят из 12 многоугольников с 476 вершинами. Скелеты фигур представляют собой геометрические графы с общим числом вершин 690 и ребер 694.

В практических задачах распознавания рукописных документов фигуры и скелеты имеют большие размеры. Например, текст на рукописной странице тотального диктанта [2], отсканированной с разрешением 300 dpi, имеет порядка 300 связных компонент, каждая из которых аппроксимируется многоугольной фигурой (рис. 13). Границы фигур состоят из 1200–1500 многоугольников, имеющих в общей сложности 60–70 тыс. вершин. Скелеты этих фигур имеют до 100 тыс. вершин и ребер. Ввиду большой размерности и разнообразия изображений рукописных документов большое значение для практического использования в архивах оцифрованных рукописей имеет реальное быстродействие и надежность работы алгоритмов.

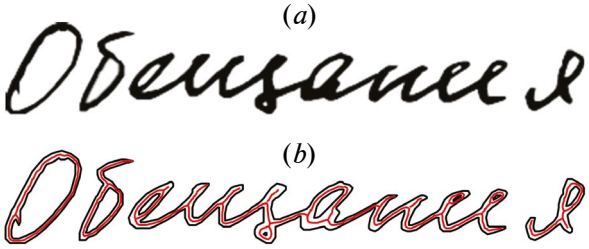

Рис. 1. (*a*) Растровое изображение текста, (*b*) многоугольные фигуры и их скелеты.

Теоретическая оценка вычислительной сложности скелетизации МФ получается на основе сведения задачи к построению обобщенной диаграммы Вороного (ДВ) множества точек и отрезков, образованных из вершин и сторон фигуры. Скелет МФ является подграфом ДВ МФ, выделение этого подграфа имеет сложность $O(n)$, где n – число вершин МФ. Для задачи построения ДВ МФ известна нижняя оценка сложности $\Omega(n \log n)$.

Известны также эффективные алгоритмы построения ДВ множества точек и отрезков, имеющие вычислительную сложность $O(n \log n)$ [3, 4]. Они основываются на разных алгоритмических парадигмах: “разделяй и властвуй” и «плоское заметание”. Эти алгоритмы носят теоретический характер, их практическая реализация на реальных процессорах, имеющих конечную точность, является серьезной проблемой. Несмотря на значительное время, прошедшее после их публикации, полные реализации этих алгоритмов неизвестны.

Одной из проблем, возникающих при построении ДВ МФ большого размера, является так называемая робастность вычислительных алгоритмов [6, 7]. Под этим термином понимается возможность получения корректного решения при наличии ошибок округления, сказывающихся на точности представления данных и результатах операций процессора. Эта проблема возникает при решении многих задач вычислительной геометрии. Она состоит в разрыве между теоретически правильными геометрическими алгоритмами и практическими компьютерными программами, выполняемыми на процессоре с конечной точностью. Фактические вычисления содержат числовые ошибки, эти ошибки вызывают несоответствия в топологии и иногда приводят к аварийному завершению программ.

При построении ДВ числовые ошибки сказываются на геометрических вычислениях предикатов, которые определяют положение точки относительно линии (прямой или окружности), и на вычислениях конструкторов, которые создают новые геометрические объекты, в частности, окружности, касающиеся трех сайтов (точек или отрезков) [7]. В обоих случаях вероятность ошибки возрастает при работе с окружностями очень большого радиуса. При создании такой окружности ищется точка пересечения прямых на основе вычисления матричного определителя. Если эти прямые “почти параллельны”, то значение определителя близко к нулю. В результате окружность имеет очень большой радиус, а центр ее расположен далеко от МФ. Небольшие ошибки вычислений приводят к неверному вычислению предикатов положения точек и отрезков относительно такой окружности. С учетом этого фактора алгоритмы, которые не требуют построения больших окружностей, являются более робастными.

Решение, предлагаемое в данной работе, основано на парадигме плоского заметания. Оно использует особенность задачи построения ДВ МФ, позволяющую избежать построения больших окружностей. Эта особенность состоит в том, что требуется найти лишь внутреннюю часть ДВ точек и отрезков, т. е. часть, лежащую внутри фигуры. А все внутренние вписанные окружности фигуры не превосходят размером саму фигуру. В алгоритмах “разделяй-и-властвуй” избежать построения больших окружностей не удается. Сравнение алгоритмов построения ДВ, основанное на этих двух парадигмах, предпринималось в работе [8]. Сравнение выполнялось по критерию вычислительной эффективности. Сравнение этих алгоритмов с точки зрения робастности не рассматривалось ранее.

Предложенное решение включает новые элементы.

1. Вводится понятие ориентированных односторонних сайтов-сегментов, для которых определена внутренняя сторона МФ. Благодаря этому появляется возможность строить только внутреннюю часть ДВ МФ. В результате не приходится строить большие окружности, а также сокращается объем вычислений, поскольку не нужно строить внешнюю часть ДВ.

2. Структура данных Статус заметающей прямой (ЗП), традиционно присутствующая в алгоритмах заметания [9], строится в виде упорядоченного множества зон влияния сайтов. Такая структура является альтернативой традиционно используемому при построении ДВ понятию волнового фронта или береговой линии.

3. В алгоритме учитывается специфика множества сайтов, образованных границей многоугольной фигуры, а именно инцидентность каждого сайта-точки двум сайтам-сегментам и каждого сайта-сегмента двум сайтам-точкам.

4. Структура данных Статус ЗП строится в виде комбинации сбалансированного дерева и двунаправленного списка, что позволяет выполнять значительную долю операций со Статусом ЗП за константное время.

Предлагаемый алгоритм построения ДВ МФ имеет существенные преимущества по сравнению с известными аналогами. Алгоритм [10]

имеет сложность $O(n\log n)$, однако он строит внутренний скелет лишь для простого n-угольника, т. е. для односвязной МФ. Для многосвязных МФ (многоугольников с многоугольными дырами) – известно решение [11], имеющее сложность $O\big(n(\log n + m)\big)$, где m – количество многоугольников-дыр. Однако для изображений рукописных документов обычно $m = O(n)$ и расходы времени составляют $O\left(n^2\right)$, что при $n \sim 10^5$ является неприемлемым. Для обеспечения практической надежности алгоритмов разрабатываются различные эвристические приемы, ориентированные на особенности конкретных программных решений [12–14]. При этом теоретическая вычислительная эффективность $O(n \log n)$ не достигается.

Известны также практические решения задачи построения ДВ точек и отрезков, которые не достигают теоретических оценок эффективности [12–16].

Предлагаемый алгоритм реализует теоретическую оценку вычислительной сложности $O(n \log n)$ и при этом имеет высокую робастность, что подтверждается вычислительными экспериментами и практическим использованием.

Оставшаяся часть статьи имеет следующую структуру. В разделе 2 дается определение внутренней диаграммы Вороного, в разделах 3–4 описывается концепция предлагаемого алгоритма. Разделы 5–7 содержат описание методов обработки событий в процессе заметания. В разделе 8 описана организация данных и связанные с нею оценки вычислительной сложности алгоритма. Описание вычислительных экспериментов представлено в разделе 9.

2. ВНУТРЕННЯЯ ДИАГРАММА ВОРОНОГО МНОГОУГОЛЬНОЙ ФИГУРЫ

Граница многоугольной фигуры состоит из одного внешнего и нескольких внутренних многоугольников. Все они описываются последовательностями своих вершин так, что внутренность МФ находится слева от границы. Вершины внешнего многоугольника упорядочены против часовой стрелки, а внутренних – по часовой стрелке. Каждый граничный многоугольник разбивается на подмножества, называемые сайтами. Вершины фигуры задают множество сайтов-точек, а стороны фигуры – множество сайтов-сегментов. На сайтах-сегментах задано направление в соответствии с направлением границы фигуры, т. е. внутренность фигуры лежит слева от сайта-сегмента.

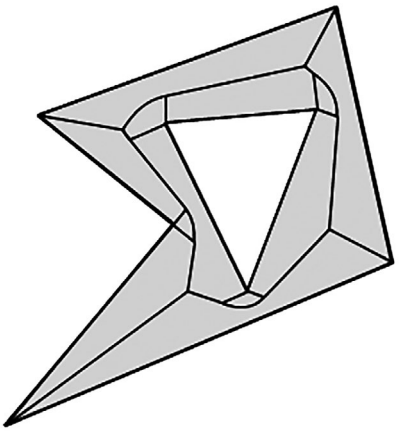

Рис. 2. Внутренняя диаграмма Вороного многоугольной фигуры.

Разбиением Вороного многоугольной фигуры будем называть представление фигуры в виде так называемых локусов. Локусом сайта называется множество точек фигуры, для которых этот сайт является ближайшим. А ближайший сайт для точки фигуры определяется положением ближайшей к ней точки на границе фигуры. Если ближайшая точка границы совпадает с вершиной фигуры, то соответствующий вершине сайт-точка является ближайшим. Если для точки фигуры ближайшей точкой границы является ее ортогональная проекция на сайт-сегмент, то этот сайт-сегмент является ближайшим. Этот ближайший сайт называется порождающим сайтом локуса.

Локус представляет собой замкнутую область. Граница локуса имеет внешнюю часть, совпадающую с границей фигуры и образованную порождающим сайтом. Остальная часть границы локуса – это разделяющая линия с соседними локусами. Граница между парой локусов называется бисектором их порождающих сайтов. Совокупность всех внутренних границ локусов многоугольной фигуры называется внутренней диаграммой Вороного этой фигуры. Внутренняя диаграмма Вороного имеет вид геометрического графа, вершинами которого являются точки фигуры, а ребрами – отрезки прямых и квадратичных парабол (рис. 2).

3. УЧАСТКИ И ЗОНЫ ЗАМЕТАЮЩЕЙ ПРЯМОЙ

Граница МФ может быть представлена в виде конечного множества монотонных ветвей. Каждая ветвь – это ломаная линия, вершины которой упорядочены лексикографически слева направо (рис. 3).

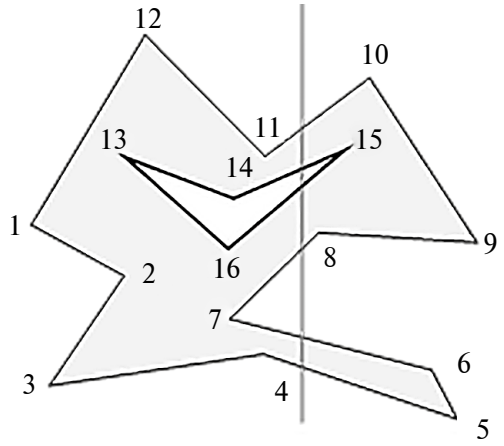

Рис. 3. Монотонные ветви границы фигуры: 1–12–11–10–9, 7–6–5, 7–8–9, 3–4–5, 1–2, 3–2, 13–14–15, 13–16–15.

Заметающая прямая (ЗП) – это вертикальная прямая линия, которая перемещается слева направо параллельно самой себе.

Ветви разбивают заметающую прямую на связные подмножества, так называемые участки. Участки внутри фигур называются внутренними, вне фигур – внешними.

Внутренние участки ЗП в свою очередь разбиваются на так называемые зоны ответственности сайтов, определяемые следующим образом. Для каждой точки внутреннего участка существует максимальный круг, лежащий внутри фигуры в левой полуплоскости относительно заметающей прямой и касающийся ее в этой точке. Поскольку круг максимальный, он также касается изнутри границы фигуры в одной или нескольких точках. Каждая точка касания принадлежит какому-то сайту. Эти сайты и круг называются инцидентными.

Зоной ответственности сайта называется отрезок заметающей прямой, все точки которого имеют касательные круги инцидентные этому сайту. Будем называть этот инцидентный сайт генератором зоны (рис. 4).

На множестве зон задано отношение порядка – снизу вверх. По мере перемещения заметающей прямой это множество зон изменяется: какие-то новые зоны на ней появляются, а какие-то зоны исчезают. Зона в момент порождения имеет длину 0, т. е. это точка на ЗП. По мере продвижения прямой вправо она превращается в отрезок, размер зоны увеличивается. Перед исчезновением размеры зоны уменьшаются до нуля, и она вырождается в точку.

Структура данных, описывающая упорядоченное множество зон на ЗП, носит название “Статус заметающей прямой”, для краткости

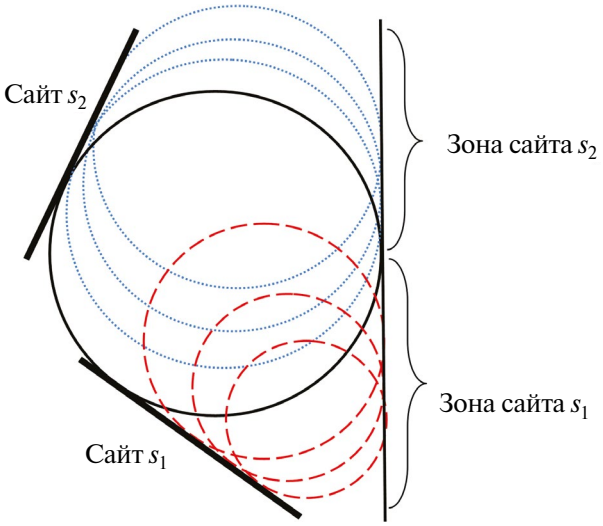

Рис. 4. Зоны сайтов-сегментов s_1, s_2. Касательные круги зоны s_1 – пунктир, зоны s_2 – точечный пунктир, общий круг двух зон – сплошная линия.

Статус, как это принято в алгоритмах плоского заметания [8].

4. ДИАГРАММА ВОРОНОГО И ПРОЦЕСС ЗАМЕТАНИЯ

По мере перемещения ЗП все зоны на ней проходят одинаковый жизненный цикл: порождение, рост и сжатие, расщепление (возможное, но не обязательное), исчезновение. Рост и сжатие представляет собой изменение размеров отрезка зоны. Сначала это монотонное увеличение, а потом уменьшение, тоже монотонное. Расщепление зоны состоит в разделении ее на две части, каждая из которых представляет собой зону с тем же самым генератором.

Между множеством зон и ДВ существует связь, которая позволяет строить ДВ в процесс заметания. Динамически изменяющееся соседство зон в Статусе ЗП полностью определяет структуру ДВ. Поэтому задача состоит в том, чтобы проследить все изменения Статуса и выявить все соседние пары зон в меняющейся структуре Статуса ЗП.

Связь между соседством зон в Статусе и ребрами и вершинами ДВ определяется на основе следующих утверждений.

1. Если два сайта имеют общий касательный вписанный круг, то найдется такое положение заметающей прямой и такая пара соседних зон на ней, у которых генераторами являются эти два сайта.

2. Если две зоны являются соседними на ЗП, то сайты-генераторы этих зон имеют смежные локусы в ДВ и общая граница этих локусов описывает ребро в ДВ.

3. Если три сайта имеют общий касательный вписанный круг, то найдется такое положение заметающей прямой и такая тройка соседних зон на ней, у которых генераторами являются эти три сайта.

Из утверждений 1 и 2 следует, что каждая пара соседних зон в Статусе задает ребро Вороного в ДВ. Отсюда вытекает, что если в процессе перемещения заметающей прямой какие-либо две зоны стали соседними, то бисектор их сайтов-генераторов образует ребро ДВ, поскольку он состоит из точек, равноудаленных от этих сайтов.

В процессе перемещения заметающей прямой соседство зон меняется только в точках событий. Изменение соседства зон в Статусе происходит при порождении новых зон и при исчезновении существующих. Порождение зон происходит, когда ЗП проходит через вершину МФ. Такое положение ЗП называется событием.

Новая зона, включенная в Статус, образует две новые соседние пары с зонами над и под ней. При исчезновении зоны и исключении ее из Статуса две зоны, расположенные над и под нею, образуют новую пару соседних зон. Эти новые пары автоматически порождают ребра ДВ.

Утверждение 3 позволяет вычислить вершины ДВ в процессе заметания. Как только какие-либо три зоны стали соседними в Статусе, нужно проверить, существует ли общий вписанный касательный круг для сайтов-генераторов этих зон.

Таким образом, для нахождения всех вершин и ребер ДВ нужно в процессе заметания отследить все пары и тройки соседних смежных зон в структуре Статус заметающей прямой.

5. ОТКРЫТИЕ ЗОН И ВЫЧИСЛЕНИЕ РЕБЕР ДИАГРАММЫ ВОРОНОГО

Общая идея процесса заметания выглядит следующим образом. Заметающая прямая перемещается слева направо дискретно, занимая положения, соответствующие моментам событий. Каждое событие приводит к изменению Статуса. Событие-вершина приводит к появлению новых зон. При этом множество зон в Статусе сохраняет упорядоченность при всех изменениях.

Изменения в Статусе выражаются в образовании новых пар и новых троек соседних зон. С каждой соседней парой зон связан соответствующий бисектор, разделяющий локусы сайтов-генераторов этих зон. А каждая тройка соседних зон соответствует вершине ДВ, являющейся центром круга, инцидентного трем сайтам-генераторам этих зон. Отслеживая изменения Статуса, можно выявить все соседние пары и тройки зон, и построить соответствующие им ребра и вершины ДВ.

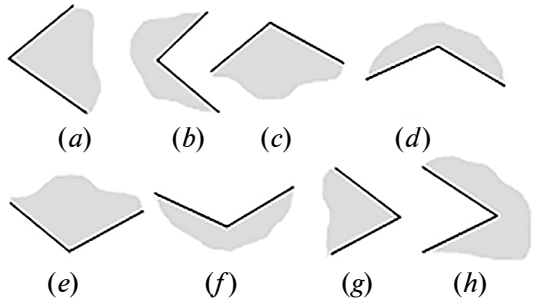

Рис. 5. Типы вершин многоугольной фигуры: левые (a, b), проходные (c, d, e, f), правые (g, h), выпуклые (a, c, e, g), вогнутые (b, d, f, h), нижние (d, e) и верхние (c, f). Внутренность многоугольной фигуры отмечена серым цветом.

Таким образом, в процессе заметания нужно для каждого события внести изменения в Статус, выявить все вновь образовавшиеся соседние пары и тройки зон, и вычислить соответствующие им вершины и ребра ДВ.

Изменения Статуса при наступлении события-вершины определяются типом вершины. Все вершины МФ разделяются на несколько типов в зависимости от пары смежных с ними сторон фигуры. Всего выделяется 8 типов вершин (рис. 5).

Обозначим:

v – сайт-точка, образованный вершиной многоугольника;

s_{pr}, s_{sc} – сайты-сегменты, образованные сторонами многоугольника, стоящими перед и после сайта-точки v в циклическом списке сайтов многоугольника;

$z(s)$ – зона, имеющая в качестве генератора сайт s;

z_* – внешняя зона, не имеющая сайта-генератора.

Событие-вершина приводит к изменению состава зон в Статусе. Эти изменения однозначно определяются в зависимости от типа вершины, с которой связано событие. Варианты этого преобразования Статуса описываются ниже для всех типов вершин, показанных на рис. 5. В строке “Вход” представлен фрагмент Статуса до наступления события, а в строке “Выход” – тот же фрагмент после события.

Левая выпуклая вершина. Сайт-точка v попадает во внешнюю зону z_*. Поскольку направление обхода многоугольника таково, что внутренность МФ лежит слева, в этом случае сайт-сегмент s_{pr} лежит выше сайта-сегмента s_{sc}. В результате обработки события зона z_* расщепляется на две внешние зоны, а между ними порождается новый

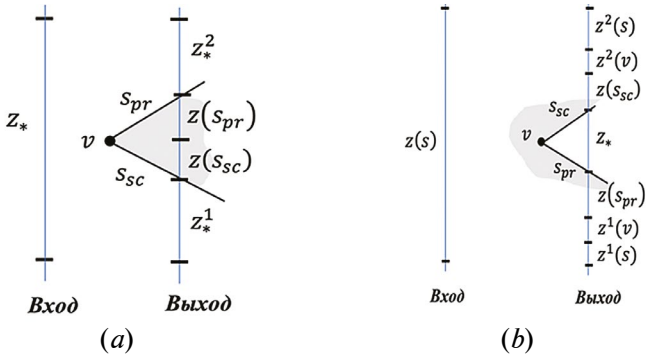

Рис. 6. Изменение Статуса при событиях “Левая вершина”: (*a*) выпуклая, (*b*) вогнутая.

внутренний участок, который состоит из двух новых зон с сайтами-генераторами s_{sc} и s_{pr}. Результат преобразования Статуса выглядит следующим образом (рис. 6*a*):

Вход: $\ldots, z_*, \ldots$

Выход: $\ldots, z_*^1, z(s_{sc}), z(s_{pr}), z_*^2, \ldots$

Образуется одна новая пара соседних зон $\{z(s_{sc}), z(s_{pr})\}$.

Левая вогнутая вершина. Сайт-точка v попадает во внутренний участок в одну из зон ЗП $z(s)$ с сайтом-генератором s. В результате обработки события зона $z(s)$ расщепляется на две зоны $z^1(s)$ и $z^2(s)$, имеющие тот же сайт-генератор s. А между $z^1(s)$ и $z^2(s)$ порождается и вставляется цепочка из пяти новых зон. Одна из них – зона z_* образована внешним участком ЗП, она лежит внутри многоугольника-дыры. Две зоны имеют сайт-генератор v и в двух зонах генераторами являются сайты-сегменты s_{pr} и s_{sc} (рис. 6*b*).

Вход: $\ldots, z(s), \ldots$

Выход: $\ldots, z^1(s), z^1(v), z(s_{pr}), z_*, z(s_{sc}), z^2(v), z^2(s), \ldots$

Здесь образуются 4 новые пары соседних зон: $\{z^1(s), z^1(v)\}$, $\{z^1(v), z(s_{pr})\}$, $\{z(s_{sc}), z^2(v)\}$, $\{z^2(v), z^2(s)\}$.

Проходная выпуклая вершина. В случае проходной вершины ЗП перед пересечением сайта-точки v пересекает инцидентный ей сайт-сегмент, лежащий левее v. Это один из соседних сайтов-сегментов s_{pr} или s_{sc} в зависимости от ориентации многоугольника. Преобразование зависит от расположения многоугольника относительно этой вершины (выше или ниже). В зависимости от этих факторов получаются два варианта преобразования множества зон.

Вершина v выпуклая верхняя (рис. 7*a*):

Вход: $\ldots, z(s_{sc}), \ldots$

Выход: $\ldots, z(s_{sc}), z(s_{pr}), \ldots$

Вершина v выпуклая нижняя (рис. 7*b*):

Вход: $\ldots, z(s_{pr}), \ldots$

Выход: $\ldots, z(s_{pr}), z(s_{sc}), \ldots$

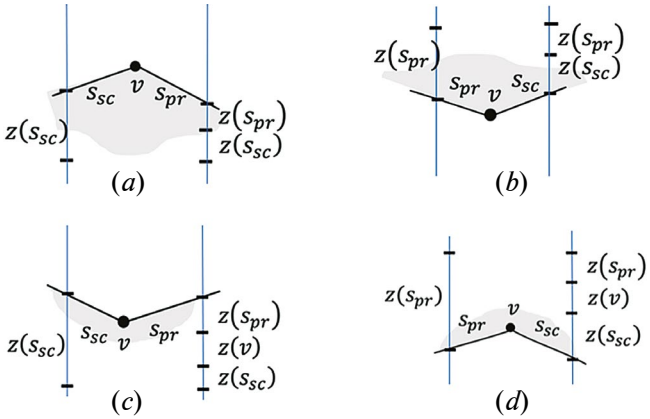

Рис. 7. Изменение Статуса при событиях “Проходная вершина”: (*a*) верхняя выпуклая, (*b*) нижняя выпуклая, (*c*) верхняя вогнутая, (*d*) нижняя вогнутая.

В обоих случаях, представленных на рис. 7*a* и 7*b*, образуется одна новая пара соседних зон $\{z(s_{pr}), z(s_{sc})\}$.

Проходная вогнутая вершина. Аналогично с рассмотренным случаем выпуклой вершины это преобразование зависит от того, находится многоугольник выше или ниже вершины. Получаются два варианта преобразования зон:

Вершина v вогнутая верхняя (рис. 7*c*):

Вход: $\ldots, z(s_{sc}), \ldots$

Выход: $\ldots, z(s_{sc}), z(v), z(s_{pr}), \ldots$

Вершина v вогнутая нижняя (рис. 7*d*):

Вход: $\ldots, z(s_{pr}), \ldots$

Выход: $\ldots, z(s_{sc}), z(v), z(s_{pr}), \ldots$

В обоих случаях на рис. 7*c* и 7*d* образуются две новые пары соседних зон: $\{z(s_{sc}), z(v)\}$, $\{z(v), z(s_{pr})\}$.

Правая выпуклая вершина. Преобразование состоит в удалении двух зон сайтов-сегментов и “сращивании” двух внешних зон (рис. 8*a*).

Вход: $\ldots, z_*^1, z(s_{pr}), z(s_{sc}), z_*^2, \ldots$

Выход: $\ldots, z_*, \ldots$

В этом случае новые соседние пары зон не образуются.

Правая вогнутая вершина. Происходит “замена” внешней зоны на зону с сайтом-генератором v (рис. 8*b*).

Вход: $\ldots, z(s_{sc}), z_*, z(s_{pr}), \ldots$

Выход: $\ldots, z(s_{sc}), z(v), z(s_{pr}), \ldots$

Образуются две новых пары соседних зон: $\{z(s_{sc}), z(v)\}$, $\{z(v), z(s_{pr})\}$.

Анализ событий-вершин показывает, что они приводят к коррекции Статуса ЗП, которая состоит в порождении до пяти новых зон. Эти зоны вставляются в Статус непосредственно одна за другой в известной последовательности. При

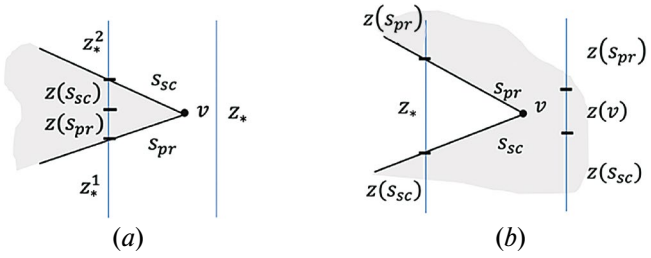

Рис. 8. Изменение Статуса при событиях “Правая вершина”: (*a*) выпуклая, (*b*) вогнутая.

этом образуется до четырех новых пар соседних зон. Каждой паре соседних зон соответствует ребро ДВ. Форма этого ребра определяется сайтами-генераторами пары соседних зон. Если это пара однотипных сайтов, т. е. они оба сайты-точки или оба сайты-сегменты, то ребро ДВ является линейным отрезком. Если же это сайт-точка и сайт-сегмент, то ребро представляет собой отрезок квадратичной параболы. Для того чтобы в явном виде построить это ребро, необходимо кроме двух сайтов-генераторов вычислить его концевые точки – вершины ДВ.

6. ЗАКРЫТИЕ ЗОН И ВЫЧИСЛЕНИЕ ВЕРШИН ДИАГРАММЫ ВОРОНОГО

В геометрическом графе, каковым является ДВ МФ, имеются вершины первой, второй и третьей степени. Вершины первой и второй степени – это точки на границе МФ. Выпуклые вершины МФ имеют в ДВ степень 1, а вогнутые – степень 2. Образование этих вершин осуществляется при наступлении события-вершина.

Вершина ДВ третьей степени – это внутренняя точка МФ, являющаяся граничной для трех локусов. Такая точка является центром вписанного круга, касающегося трех сайтов. Вычисление этих вершин ДВ также выполняется в процессе заметания. Однако их появление связано с другим типом событий – это события-круги.

Новая зона после порождения помещается в Статус ЗП, где она может образовать с другими зонами новые тройки соседних зон. Количество таких новых троек может составить от 0 до 3. Если для сайтов-генераторов тройки соседних зон существует касательный круг, то его центр может оказаться точкой вершиной ДВ. Но для этого нужно, чтобы круг этот был пустым. А достоверно установить его пустоту можно будет лишь в тот момент, когда ЗП при своем движении займет положение касательной к кругу справа, пройдя его полностью. Соответствующее событие, когда ЗП станет касательной к кругу справа, называется событием-кругом. С этим событием связано порождение новой вершины ДВ и удаление из Статуса одной зоны – средней из тройки соседних зон.

Таким образом, для нахождения всех вершин Вороного в процессе заметания необходимо выполнить следующие действия.

При порождении новой зоны в Статусе нужно проверить условие существования общих касательных кругов для троек сайтов-генераторов образовавшихся новых троек соседних зон. Если такой круг существует, то точка центр этого круга является кандидатом на объявление ее вершиной ДВ. Для этого круга нужно запланировать событие-круг. Далее при наступлении этого события точка центр круга объявляется вершиной ДВ.

Событие-круг приводит к исчезновению и исключению из Статуса одной зоны. Удаляется средняя зона из тройки, определяющей круг.

Иллюстрация последовательности событий и изменений, происходящих в Статусе, представлена на рис. 9. Здесь фигура – четырехугольник с сайтами-сегментами s_1, s_2, s_3, s_4. События-вершины связаны с положениями ЗП, обозначенными A, B, C, G.

В положении B для тройки соседних зон $z(s_1)$, $z(s_2)$, $z(s_3)$ образуется вписанный круг сайтов-генераторов s_1, s_2, s_3. Этот круг изображен жирной линией. При его порождении создается событие-круг, привязанное к позиции F. Далее в положении ЗП C в Статус вводится зона $z(s_4)$, в результате чего образуется новая тройка соседних зон $z(s_2)$, $z(s_3)$, $z(s_4)$. Для сайтов s_2, s_3, s_4 существует вписанный круг, для которого создается событие-круг D. Это событие привязано к средней зоне

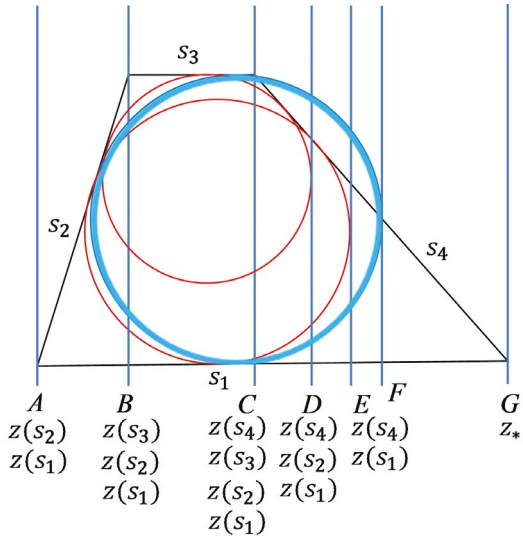

Рис. 9. Заметание четырехугольника, события вершины – A, B, C, G, события-круги D, E, F, изменения Статуса по событиям.

тройки $z(s_3)$. При наступлении события D зона $z(s_3)$ вырождается в точку и удаляется из Статуса.

После ее удаления появляется новая тройка соседних зон $z(s_1)$, $z(s_2)$, $z(s_4)$. Для сайтов-генераторов строится круг, который порождает событие-круг в момент E. Это событие привязано к средней зоне тройки $z(s_2)$. Поскольку эта зона ранее была привязана к кругу, который больше не имеет контакта с ЗП, а лежит левее ее, производится перепланирование событий. Событие F удаляется вместе с привязанным к нему кругом, а для зоны $z(s_2)$ планируется новое событие-круг E.

События-вершины и события-круги полностью описывают процесс получения ребер и вершин ДВ при заметании.

7. ВЫЧИСЛЕНИЕ КАСАТЕЛЬНЫХ КРУГОВ

Задача вычисления координат вершин ДВ возникает при использовании любых алгоритмов. Эта задача имеет чисто геометрическую природу. Нужно построить касательную окружность для трех сайтов, причем касание должно быть в заданной последовательности. Возникает шесть возможных геометрических задач на построение. Это разнообразие определяется составом тройки сайтов (3 точки, 2 точки и отрезок, точка и 2 отрезка, 3 отрезка), а также особыми случаями, когда точка совпадает с концом отрезка (рис. 10).

Поскольку внутренняя вершина ДВ всегда есть точка пересечения пары бисекторов, для ее нахождения могут быть использованы уравнения бисекторов сайтов [17]. Пересечение бисекторов находится на основании решения системы из двух уравнений первой или второй степени. Полученное решение этой системы нуждается в постобработке, поскольку оно может оказаться геометрически некорректным. Кроме того, решение может быть не единственным и необходимо выбрать один вариант из нескольких.

Похожие задачи возникают при определении границ зон. Здесь нужно строить касательные

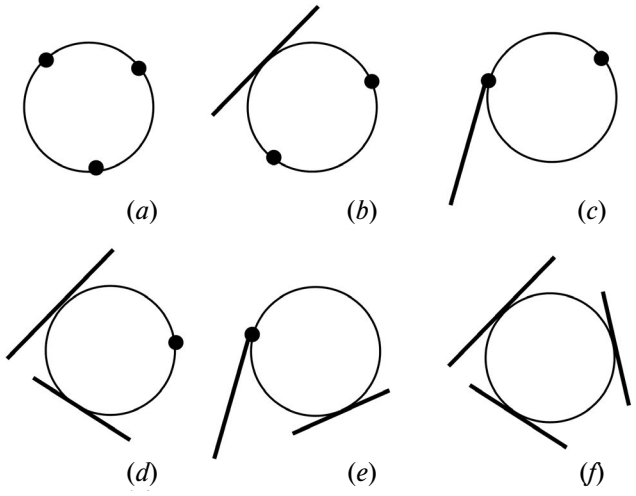

Рис. 10. Геометрические задачи построения касательной окружности для точек и отрезков.

окружности для двух сайтов и заметающей прямой (рис. 11).

Для решения этих задач мы применили методы аналитической геометрии, которые предоставляют ясную геометрическую интерпретацию результатов на любом этапе вычислений. В основе алгоритмов лежит использование классических методов решения геометрических задач на построение, т. е. с помощью циркуля и линейки.

8. ВЫБОР СТРУКТУР ДАННЫХ И ТЕОРЕТИЧЕСКАЯ СЛОЖНОСТЬ

Алгоритмическая парадигма плоского заметания реализуется в виде двух структур данных, описывающих перечень событий и Статус ЗП [9]. В предлагаемом алгоритме перечень событий представлен в виде структуры данных “очередь с приоритетами”, которая эффективно реализуется с помощью АВЛ-дерева. Перечень событий включает события-вершины и события-круги, общее число которых для МФ с n вершинами оценивается как $O(n)$. Для работы с перечнем событий требуются операции вставки, удаления, поиска события и определения самого первого

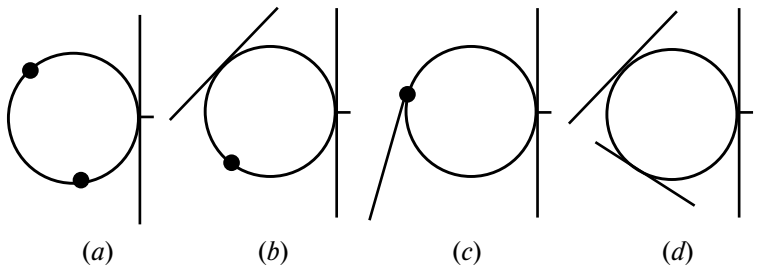

Рис. 11. Вычисление касательного круга для двух сайтов и заметающей прямой: (*a*) –сайты-точки, (*b*, *c*) – сайт-точка и сайт-сегмент, (*d*) – два сайта-сегмента.

события среди оставшихся. Сложность каждой из этих операций в АВЛ-дереве составляет $O(\log n)$.

Статус ЗП должен поддерживать операции, используемые в структуре Словарь: вставка, удаление, поиск, под и над. Как известно, при использовании для реализации словаря АВЛ-дерева, все эти операции также можно выполнять за время $O(\log m)$, где m – число элементов в словаре. В нашем случае элементами являются зоны сайтов. Общее количество зон, образованных в процессе заметания, составляет величину $O(n)$. Если реализовать эту структуру также в виде АВЛ-дерева, то получится эффективное решение, в котором сложность выполнения перечисленных операций составит $O(\log n)$.

Однако особенности рассматриваемой задачи построения ДВ МФ требуют разработки специальной структуры для Статуса ЗП.

Первая особенность состоит в том, что вставка зон в Статус осуществляется во многих случаях целыми пакетами. Как видно из проведенного анализа событий-вершин, вставляется цепочка от 1 до 5 зон одновременно. Это значит, что нерационально использование механизма АВЛ-дерева, где каждая вставка занимает логарифмическое время. Нужно найти способ, при котором эта вставка сможет за один заход вставить весь пакет зон.

Вторая особенность связана с тем, что все зоны в одном пакете в начальный момент при порождении имеют нулевые размеры, т. е. это точки на ЗП. При этом точки эти имеют одну и ту же ординату, т. е. их положение на ЗП просто совпадает. Для того чтобы вводить зоны последовательно по одной с помощью АВЛ-дерева нужно иметь какой-то способ сравнения этих зон по критерию выше-ниже. Но стандартная операция вставки элементов в АВЛ-дерево не предоставляет таких средств.

Обойти проблемы, связанные с этими особенностями предлагается с помощью комбинированной структуры данных, включающей АВЛ-дерево и двунаправленный список.

Двунаправленный список представляет собой упорядоченное снизу вверх множество зон. При этом внешние участки ЗП представлены как отдельные зоны в том же формате, что и зоны внутренних участков. Главное достоинство представления Статуса таким образом состоит в том, что большая часть операций вставки зон при наступлении событий-вершин, описанных в разделе “Открытие зон”, может быть выполнена за константное время. События, связанные с прохождением ЗП через проходные и правые вершины, состоят в порождении нескольких зон и вставке их в заданном порядке в Статус. При этом место вставки известно, оно определяется зоной сайта-сегмента, инцидентного вершине и лежащего левее ее в монотонной ветви. Таким образом, такая вставка зон в количестве до 5 не зависит от числа зон в Статусе и выполняется за константное время.

Единственное событие, обработка которого не укладывается в этот простой механизм – это “левая вершина”. Для вставки левой вершины место в Статусе заранее неизвестно, поэтому требуется локализация ее на ЗП. Это значит, что нужно найти зону, в которую попадает левая вершина. Двунаправленный список дает возможность это сделать только путем последовательного просмотра зон. Такой просмотр имеет сложность $O(n)$ в худшем случае. Поскольку число левых вершин в МФ имеет порядок $O(n)$, получается, что операции по вставке зон таких вершин в Статус имеют сложность $O(n^2)$, что является неприемлемым.

Предлагаемое решение состоит в том, чтобы дополнить двунаправленный список зон структурой АВЛ-дерева, описывающей упорядоченное снизу вверх множество ветвей, пересекающих ЗП. Эта структура позволяет определить для точки на ЗП пару ветвей, лежащих выше и ниже точки. Такая пара ветвей задает участок на ЗП. Каждая ветвь содержит ссылки на зоны, прилегающие к ней при текущем положении ЗП. Эта конструкция обеспечивает локализацию точки в множестве участков за время $O(\log n)$, а локализацию всех левых вершин – $O(n \log n)$.

Если найденный участок оказался внешним, то никаких дополнительных действий не требуется и вставка новой группы зон выполняется по правилу, представленному на рис. 6*а*. Если же найденный участок является внутренним, то для дальнейшего поиска зоны предлагается алгоритм, который можно назвать “вилкой”.

Предположим, что найденный внутренний участок содержит k зон $z_1, \ldots, z_k$. Пара (z_{low}, z_{up}) задает нижнюю и верхнюю зоны интервала поиска. Вначале $z_{low} = z_1$, $z_{up} = z_k$. Обозначим также $z.above$ и $z.under$ зоны в Статусе, расположенные выше и ниже зоны z.

Поиск зоны, содержащей левую вершину v, осуществляется итеративно, каждая итерация включает две проверки:

• если вершина v локализуется в зоне z_{low}, возвращается z_{low}, иначе интервал поиска сжимается снизу, $z_{low} = z_{low}.above$;

• если вершина v локализуется в зоне z_{up}, возвращается z_{up}, иначе интервал поиска сжимается сверху, $z_{up} = z_{up}.under$.

Этот итерационный процесс гарантированно закончится успешно, поскольку вершина v обязательно попадает в одну из зон на участке. Вычислительная сложность при этом составит $O(k)$. При этом максимальное число проверок k потребуется в том случае, когда вершина v лежит в медианной зоне z_t, $t = \left\lceil \frac{k}{2} \right\rceil$, занимающей серединное положение в участке.

Покажем, что локализация всех левых вершин в зонах ЗП имеет сложность $O(n\log n)$. Число зон, порождаемых в процессе заметания, составляет $O(n)$. Без ограничения общности будем считать, что зон ровно n. Обозначим m – число левых вершин МФ, $m = O(n)$. Оценим максимальное число проверок за все время работы. При наступлении события “левая вершина” максимальное число проверок потребуется при выполнении двух условий:

1. Вершина локализуется в участке с наибольшим числом зон.

2. При локализации на участке вершина попадает в медианную зону, т. е. совершается k проверок, где k – число зон на участке.

Таким образом, если всегда выполняется максимальное число проверок, то для первой левой вершины в участке из n зон совершается n проверок. После образования новых зон, связанных с этой вершиной, образуются два новых участка, которые будут содержать по $\frac{n}{2}$ зон. Две последующие локализации с максимальным числом проверок в участках длины $\frac{n}{2}$ потребуют по $\frac{n}{2}$ проверок каждый. После вставки соответствующих зон в Статус образуются четыре участка по $\frac{n}{4}$ зон, и т. д. На каждом уровне j образуется 2^{j-1} участков, включающих по $\frac{n}{2^{j-1}}$ зон каждый. На этом уровне происходит 2^{j-1} локализаций левых вершин с суммарным числом проверок n. За j уровней обрабатывается $1 + 2 + 4 + \ldots + 2^{j-1} = 2^j - 1$ событий типа “левая вершина”. При $j = \log(m+1)$ получаем, что все левые вершины будут обработаны за j уровней. А так как на каждом уровне всего n проверок, получаем общее число проверок $O(n\log m)$. Так как $m = O(n)$, получаем итоговую сложность $O(n\log n)$.

Таким образом, предложенный алгоритм построения ДВ многоугольной фигуры с использованием двухступенчатого иерархического Статуса имеет асимптотическую сложность $O(n\log n)$, где n – суммарное число вершин в МФ.

9. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

Для практической проверки алгоритма на корректность, эффективность и надежность используется набор изображений различной сложности. Сложность определяется числом многоугольников и вершин в МФ, полученных при аппроксимации границ изображения.

Проверка эффективности проводилась путем сравнения времени работы предложенного алгоритма с реализацией алгоритма Форчуна из библиотеки C++ Boost [18]. Данная реализация включает построение обобщенной ДВ для произвольного набора точек и отрезков на плоскости. То есть алгоритм в библиотеке Boost никак не учитывает взаимное расположение точек и сегментов в виде многоугольников, а также строит разбиение всей плоскости, а не только внутренней части МФ.

После построения ДВ всей плоскости алгоритмом Форчуна необходимо провести постобработку, состоящую в удалении внешних относительно МФ ребер ДВ. Однако в экспериментах время, затрачиваемое на постобработку в алгоритме Форчуна, не учитывается. Таким образом, цифры затрат времени для библиотеки Boost представляют собой нижние оценки для построения ДВ МФ, в них не включено время постобработки. Результаты временных замеров приведены в табл. 1. Примеры некоторых изображений, использованных в экспериментах, (Лошадь, Нейрон, Дерево, План) представлены на рис. 12.

Сравнение двух алгоритмов проводилось в единой среде при одинаковых условиях, время усреднялось по 10 замерам. Тем не менее абсолютное время работы программы в значительной степени есть характеристика реализации, а не самого алгоритма. Более показательным в данном эксперименте является относительное изменение времени работы на изображениях разного размера. На картинках небольшого размера (до 5000 вершин МФ) алгоритм из Boost проигрывает нашему алгоритму в 2–4 раза, на больших же картинках, с более чем 100 тысячами вершин, время работы алгоритмов отличается примерно в 50 раз.

Результаты еще одного эксперимента приведены в табл. 2. Здесь представлены результаты

Таблица 1. Вычисление ДВ тестовых примеров многоугольных фигур

Картинка	Число вершин	Число контуров	Время предложенного алгоритма, мс (T_1)	Время алгоритма из библиотеки Boost, мс (T_2)	Отношение расходов времени (T_2/T_1)
Лошадь	374	1	2.6	3.5	1.35
Нейрон	3449	7	25.5	60	2.35
Дерево	175375	6373	1894	84537	44.63
План	185115	5395	1659	94908	57.20

Рис. 12. Тестовые примеры многоугольных фигур: лошадь (1 многоугольник, 374 вершины), нейрон (7, 3449), дерево (6373, 175375), план (5395, 185115).

Таблица 2. Вычисление ДВ фигур изображений рукописного текста

Документ	Количество контуров	Количество вершин в контурах	Количество вершин в ДВ	Время, мс
1	917	27991	54824	253
2	1710	85938	104558	914
3	1332	100040	108427	1106
4	1696	112746	131366	1255
5	1470	106961	115050	1175

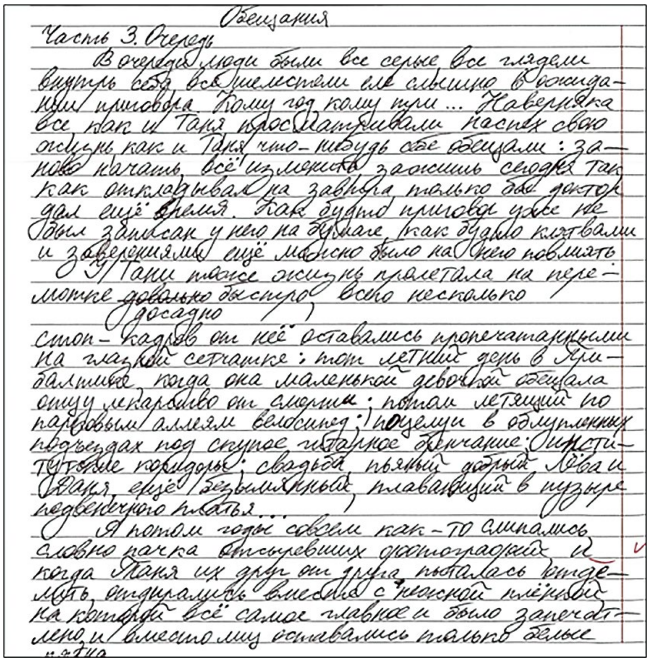

Рис. 13. Страница тотального диктанта – пример множества многоугольных фигур, аппроксимирующих изображение рукописного текста.

работы алгоритма с пятью цифровыми изображениями рукописного текста. Пример такого изображения представлен на рис. 13. Исходные бинарные растровые изображения аппроксимируются МФ с помощью алгоритмов [1]. Размеры получаемых фигур приведены в таблице. Затраты времени на построение ДВ выражены в миллисекундах.

Этот эксперимент показал применимость алгоритма к решению практических задач обработки больших изображений рукописных текстов размера 4000×2500 пикселов. Полученные ДВ используются для получения непрерывного скелета и на его основе построения штриховой сегментации рукописи. Штриховая сегментация находит применение в алгоритмах распознавания рукописного текста при решении задач расшифровки текста, навигации в больших рукописных архивах, идентификации авторов.

10. ЗАКЛЮЧЕНИЕ

В статье представлен алгоритм построения внутренней ДВ МФ, ориентированный на практическую программную реализацию и приложения к задачам большого размера, в частности, на работу с большими изображениями рукописных документов. За счет того, что ДВ строится только для внутренней части МФ, достигается несколько полезных свойств алгоритма, способствующих повышению вычислительной эффективности и числовой надежности.

Основная концепция алгоритма основана на парадигме плоского заметания, использованной в алгоритме Форчуна. Предложенный алгоритм реализует редукцию задачи к построению ДВ линейных отрезков, но при этом включает новые элементы, ориентированные на использование свойств отрезков, составленных из многоугольников. За счет этого достигается уменьшение вычислительной сложности, поскольку значительная часть операций, которые в классическом алгоритме Форчуна имеют логарифмическую сложность, реализуется за константное время. Кроме того, сокращается объем вычислений по сравнению с известными алгоритмами, которые строят внутреннюю и внешнюю части ДВ МФ.

Предложенный алгоритм обладает высокой численной надежностью, поскольку внутренняя ДВ МФ не требует вычислений вписанных кругов большого размера, а также нахождения вершин ДВ, расположенных на очень большом расстоянии от фигуры.

Предложенный алгоритм реализован в полном объеме и используется при решении практических задач анализа и распознавания цифровых изображений рукописных текстов.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке Российского научного фонда, грант № 22-68-00066, https://rscf.ru/project/22-68-00066/.

СПИСОК ЛИТЕРАТУРЫ

1. *Местецкий Л.М.* Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры. М.: Физматлит, 2009.
2. *Отургашева Н.В.* Послание URBI ET ORBI: тотальный диктант как культурный проект // Вестник Томского государственного университета. 2019. № 35. С. 105–113. https://doi.org/10.17223/22220836/35/10
3. *Fortune S.* A sweepline algorithm for Voronoi diagrams. Algorithmica. 2 (1987). P. 153–174.
4. *Yap C.K.* An O(n log n) algorithm for the Voronoi diagram of the set of simple curve segments. Discrete Comput. Geom. 2 (1987). P. 365–393.
5. *Местецкий Л.М.* Скелетизация многосвязной многоугольной фигуры на основе дерева смежности ее границы // Сиб. журн. вычисл. матем. 2006. Т. 9. № 3. С. 299–314.
6. *Fortune S.* (1996). Robustness issues in geometric algorithms. In: Lin, M.C., Manocha, D. (eds) Applied Computational Geometry Towards Geometric Engineering. WACG 1996. Lecture Notes in Computer Science. V. 1148. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0014476
7. *Shewchuk J.R.* (2013). Lecture Notes on Geometric Robustness. University of California at Berkeley, Berkeley, CA 94720.
8. *Лагно Д., Соболев А.* Модифицированные алгоритмы Форчуна и Ли скелетизации многоугольной фигуры. Графикон-2001, Н. Новгород.
9. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение: Пер. с англ. – М.: Мир, 1989. – 478 с.
10. *Lee D.T.* Medial axes transform of planar shape // IEEE Trans. Patt. Anal. Mach. Intell. PAMI-4. 1982. P. 363–369.
11. *Srinivasan V., Nackman L.R.* Voronoi diagram for multiply-connected polygonal domains I: Algorithm // In IBM Journal of Research and Development, V. 31. No. 3. P. 361–372. May 1987. https://doi.org/10.1147/rd.313.0361.
12. *Held M.* Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments // Computational Geometry. 2001. V. 18. P. 95–123.

13. *Sugihara K., Iri M., Inagaki H. et al.* Topology-Oriented Implementation – An Approach to Robust Geometric Algorithms. Algorithmica 27, 5–20 (2000). https://doi.org/10.1007/s004530010002
14. *Karavelas M.I.* A robust and efficient implementation for the segment Voronoi diagram. In Proc. Internat. Symp. on Voronoi diagrams in Science and Engineering (VD2004), 2004. P. 51–62.
15. *Imai T.* A topology oriented algorithm for the voronoi diagram of polygons. cccg1996, 1996. P. 107–112.
16. *Bae, S.W.* (2015). An Almost Optimal Algorithm for Voronoi Diagrams of Non-disjoint Line Segments. In: Rahman M.S., Tomita E. (eds) WALCOM: Algorithms and Computation. WALCOM 2015. Lecture Notes in Computer Science. V. 8973. Springer, Cham. P. 34–43.
17. *Marsden D.* Exact Generalized Voronoi Diagram Computation using a Sweepline Algorithm (2020). All Graduate Theses and Dissertations. 7947. https://digitalcommons.usu.edu/etd/7947
18. https://www.boost.org/doc/libs/1_59_0/libs/polygon/doc/voronoi_main.htm

CONSTRUCTING THE INTERNAL VORONOI DIAGRAM OF A POLYGONAL FIGURE USING THE SWEEP METHOD

© 2024 L. M. Mestetskiy[a, b], D. A. Koptelov[a]

[a] *Lomonosov Moscow State University*
Leninskie Gory 1, GSP-1, Moscow, 119991 Russia
[b] *National Research University Higher School of Economics*
Pokrovsky Boulevard 11, Moscow, 109028 Russia

The article considers the problem of constructing the internal Voronoi diagram of a polygonal figure – a polygon with polygonal holes. A method based on the flat sweeping paradigm is proposed. Direct construction of only the internal part of the Voronoi diagram allows us to reduce the amount of calculations and increase robustness compared to known solutions. Another factor in reducing computational complexity is the use of the property of pairwise incidence of linear segments formed by the sides of a polygonal figure. To take these features into account, it is proposed to build the data structure Status of the sweeping line in the form of an ordered set of sites' areas of responsibility. The structure is implemented as a combination of a balanced tree and a bidirectional list. Computational experiments illustrate the numerical reliability and efficiency of the proposed method.

Keywords: Voronoi diagram

REFERENSES

1. *Mestetskiy L.M.* Continuous morphology of binary images: figures, skeletons, circulars. Moscow, Fizmatlit, 2009 (in Russian).
2. *Oturgasheva N.V.* URBI ET ORBI message: total dictation. as a cultural project. Bulletin of Tomsk State University. 2019. No. 35. P. 105–113 (in Russian). https://doi.org/10.17223/22220836/35/10
3. *Fortune S.* A sweepline algorithm for Voronoi diagrams. Algorithmica. 2 (1987). P. 153–174.
4. *Yap C.K.* An O(n log n) algorithm for the Voronoi diagram of the set of simple curve segments. Discrete Comput. Geom. 2 (1987). P. 365–393.
5. *Mestetskiy L.M.* Skeletonization of a multiconnected polygonal figure based on the adjacency tree of its boundary // Sibirsk. magazine Comput. Mat. 2006. V. 9. No. 3. P. 299–314 (in Russian).
6. *Fortune, S.* (1996). Robustness issues in geometric algorithms. In: Lin, M.C., Manocha, D. (eds) Applied Computational Geometry Towards Geometric Engineering. WACG 1996. Lecture Notes in Computer Science. V. 1148. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0014476
7. *Shewchuk J.R.* (2013). Lecture Notes on Geometric Robustness. University of California at Berkeley, Berkeley, CA 94720.
8. *Lagno D., Sobolev A.* Modified Fortune and Lee algorithms for skeletonization of a polygonal figure. Graphicon-2001, Nizhny Novgorod (in Russian).
9. *Preparata F., Sheimos M.* Computational geometry: Introduction: Transl. from English. – M.: Mir, 1989. – 478 p.
10. *Lee D.T.* Medial axes transform of planar shape // IEEE Trans. Patt. Anal. Mach. Intell. PAMI-4. 1982. P. 363–369.
11. *Srinivasan V., Nackman L.R.* Voronoi diagram for multiply-connected polygonal domains I: Algorithm // In IBM Journal of Research and Development, V. 31. No. 3. P. 361–372. May 1987. https://doi.org/10.1147/rd.313.0361.
12. *Held M.* Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments // Computational Geometry. 2001. V. 18. P. 95–123.
13. *Sugihara K., Iri M., Inagaki H. et al.* Topology-Oriented Implementation – An Approach to Robust Geometric

Algorithms. Algorithmica 27, 5–20 (2000). https://doi.org/10.1007/s004530010002

14. *Karavelas M.I.* A robust and efficient implementation for the segment Voronoi diagram. In Proc. Internat. Symp. on Voronoi diagrams in Science and Engineering (VD2004), 2004. P. 51–62.
15. *Imai T.* A topology oriented algorithm for the voronoi diagram of polygons. cccg1996, 1996. P. 107–112.
16. *Bae, S.W.* (2015). An Almost Optimal Algorithm for Voronoi Diagrams of Non-disjoint Line Segments. In: Rahman M.S., Tomita E. (eds) WALCOM: Algorithms and Computation. WALCOM 2015. Lecture Notes in Computer Science. V. 8973. Springer, Cham. P. 34–43.
17. *Marsden D.* Exact Generalized Voronoi Diagram Computation using a Sweepline Algorithm (2020). All Graduate Theses and Dissertations. 7947. https://digitalcommons.usu.edu/etd/7947
18. https://www.boost.org/doc/libs/1_59_0/libs/polygon/doc/voronoi_main.htm