ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2025 Управление, вычислительная техника и информатика № 71
Tomsk State University Journal of Control and Computer Science

Научная статья
УДК 515.142.33
doi: 10.17223/19988605/71/9

Быстрая реализация алгоритма невыпуклого полосового слияния триангуляции Делоне

Марина Игоревна Литовченко[1], Юрий Леонидович Костюк[2]

[1, 2] Национальный исследовательский Томский государственный университет, Томск, Россия
[1] litovchenkom21@yandex.ru
[2] kostyuk.yu48@mail.ru

Аннотация. Рассматривается быстрая реализация полосового алгоритма триангуляции Делоне набора точек плоскости. Вначале область с точками делится на полосы некоторой ширины, и производится сортировка точек в каждой полосе. Далее в каждой полосе строится отдельная невыпуклая триангуляция методом заметающей прямой. Затем соседние полосы сшиваются между собой, а после получения триангуляции всей области выполняется перестроение треугольников по критерию Делоне. Приведены результаты работы алгоритма для равномерного и неравномерного (фрактального) распределения точек внутри области. Время работы предложенного алгоритма по сравнению с наиболее быстрым алгоритмом динамического кэширования оказалось меньше в 1,7 раза. Это объясняется существенно меньшим количеством проверок и перестроений треугольников по критерию Делоне в предложенном алгоритме.

Ключевые слова: триангуляция Делоне; полосовой алгоритм; алгоритм заметающей прямой; сравнение алгоритмов.

***Для цитирования*:** Литовченко М.И., Костюк Ю.Л. Быстрая реализация алгоритма невыпуклого полосового слияния триангуляции Делоне // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2025. № 71. С. 93–102. doi: 10.17223/19988605/71/9

Original article
doi: 10.17223/19988605/71/9

Fast implementation of a non-convex stripe merging algorithm for Delaunay triangulation

Marina I. Litovchenko[1], Yuriy L. Kostyuk[2]

[1, 2] National Research Tomsk State University, Tomsk, Russian Federation
[1] litovchenkom21@yandex.ru
[2] kostyuk.yu48@mail.ru

Abstract. The paper presents a fast implementation of a Stripe Merging algorithm for Delaunay triangulation of a set of points in the plane. Initially, the area containing the points is divided into stripes of a specified width, and the points within each stripe are sorted. Subsequently, a separate non-convex triangulation is constructed in each stripe using the Sweep Line algorithm. Neighboring stripes are then merged together, and upon obtaining a triangulation of the entire area, the triangles are rebuilt according to the Delaunay criterion. The performance of the algorithm is demonstrated on both uniform and non-uniform (fractal) point distributions within the domain. The execution time of the proposed algorithm turned out to be 1,7 times faster than that of the most efficient iterative algorithm with dynamic hashing for triangle searches. This improvement can be attributed to the significantly reduced number of Delaunay criterion checks and triangles rebuilds performed by the proposed method.

Keywords: Delaunay triangulation; stripe merging algorithm; sweep line algorithm; algorithm comparison.

***For citation*:** Litovchenko, M.I., Kostyuk, Yu.L. (2025) Fast implementation of a non-convex Stripe Merging algorithm for Delaunay triangulation. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika – Tomsk State University Journal of Control and Computer Science*. 71. pp. 93–102. doi: 10.17223/19988605/71/9

Введение

Задача построения триангуляции по набору заданных точек является одной из самых трудоемких задач, которые повсеместно используются в различных областях. Например, для распознавания объектов и построения их поверхностей при обработке данных лазерного сканирования [1, 2], для идентификации пространственной сцены в системе технического зрения робота [3], при моделировании местности и распознавании образов [4].

Классический алгоритм триангуляции Делоне для точек на плоскости по методу «разделяй и властвуй» имеет порядок трудоемкости в наихудшем случае $O(n \cdot \log n)$ [5]. На данный момент создано достаточно много эффективных реализаций алгоритмов для триангуляции [6], один из самых эффективных – алгоритм динамического кэширования поиска треугольников со средней трудоемкостью $O(n)$. Тем не менее исследования по разработке новых подходов или модификации уже существующих для определенных задач остаются актуальными. Обзор основных современных алгоритмических подходов можно найти в [7].

Модификация алгоритма заметающей прямой изложена в [8], основная идея заключается в учете свойств описанной вокруг треугольника окружности, что позволяет уменьшить количество вытянутых треугольников.

Например, в работе [9] описывается альтернативный подход для построения триангуляции Делоне – генетический алгоритм, который вычисляет приближенную оптимальную триангуляцию, но работает дольше традиционных алгоритмов. Этот алгоритм позволяет динамически добавлять новые точки в триангуляцию, что может потребоваться в ряде задач. Кроме того, он может быть распараллелен.

Не каждый алгоритм триангуляции можно реализовать параллельно, поэтому подобные исследования также являются актуальными. Однако из-за вычислительной сложности некоторых этапов алгоритмов их последовательная реализация иногда может отрабатывать быстрее параллельной. Так, в работе [10] последовательная реализация триангуляции произвольных областей методом «разделяй и властвуй» [5] оказалась эффективнее параллельной.

Одним из наиболее подходящих алгоритмов для исследования возможности распараллеливания является алгоритм полосового слияния [11].

В данной работе представлена такая модификация алгоритма полосового слияния, которая позволила его существенно ускорить по сравнению с алгоритмом динамического кэширования даже при последовательной реализации.

1. Общая схема алгоритма полосового слияния

Алгоритм полосового слияния содержит следующие этапы:

1) разбиение рабочей области с точками на полосы, например по координате *Y*;

2) сортировка точек по координате *X* в каждой полосе;

3) триангуляция по точкам внутри каждой полосы методом заметающей прямой;

4) сшивание полученных триангуляций в одну;

5) проверка и перестроение всех треугольников по критерию Делоне [6].

При реализации алгоритма в работе [11] были обнаружены следующие проблемы:

1) на этапе триангуляции внутри полосы могли создаваться вырожденные треугольники, когда все три точки лежат на одной прямой;

2) при сшивании двух полос из-за невыпуклых границ триангуляций в полосах затруднительно создавать корректные треугольники.

Первая проблема в работе [11] решалась добавлением в алгоритм дополнительных проверок и перестроений, а вторая – достраиванием границ полос в одном варианте до полной выпуклости, а в другом варианте – до частичной выпуклости с дополнением многочисленных проверок. В результате появлялось большое количество вытянутых треугольников, которые позже снова перестраивались

по критерию Делоне. Из-за этого оба варианта алгоритма существенно проигрывали по быстродействию алгоритму динамического кэширования [11].

В данной работе предлагается ряд изменений на 1-м, 3-м и 4-м этапах работы алгоритма, позволивших значительно ускорить его выполнение.

2. Обработка точек и структуры данных

Рабочая область для построения триангуляции делится на некоторое количество полос одинаковой ширины. Ширина полос подбирается таким образом, чтобы обеспечить минимум перестроений треугольников: стороны треугольников должны быть близки к средней величине. Оптимальное количество полос, приведенное в работе [11], можно рассчитать по упрощенной формуле (с округлением до целого)

$$m = \left\lfloor 0{,}5\sqrt{\frac{b}{a}n} + 0{,}5 \right\rfloor, \tag{1}$$

где m – количество полос, a – размер области по координате X, b – размер области по координате Y, n – количество точек в области.

Для каждой полосы формируется список входящих точек, а также дополнительные четыре граничные точки. Стоит отметить, что координаты всех точек преобразованы к целым числам, а минимальные и максимальные границы рабочей области – $x_{\min}$, $x_{\max}$ по координате X, $y_{\min}$, $y_{\max}$ по координате Y – расширяются на 1 по всем направлениям. Таким образом, для всех граничных точек следует вычислить значения только по координате Y, так как по координате X значения будут равны соответственно $x_{\min} - 1$, $x_{\max} + 1$. Значения по координате Y для граничных точек вычисляются по следующему рекуррентному соотношению:

$$\begin{cases} P_1 = y_{\min} - 1;\ S_1 = P_1; \\ P_i = P_{i-1} + p;\ S_i = \lfloor P_i \rfloor,\ i = \overline{2,m}; \\ S_{m+1} = y_{\max} + 1; \end{cases} \tag{2}$$

где $p = (y_{\max} - y_{\min} + 2)/m$ – средняя ширина полосы, P_i – вещественная переменная, S_i – нижняя граница по оси Y для i-й полосы. Тогда в i-ю полосу попадают точки с координатами по оси Y: $S_i \le y < S_{i+1}$.

Важным этапом является сортировка точек по оси X для каждой полосы, а при одинаковых координатах X – по оси Y, что гарантирует отсутствие вырожденных треугольников при построении невыпуклой триангуляции методом заметающей прямой в каждой полосе. При использовании метода цифровой сортировки можно добиться линейного порядка трудоемкости $O(n)$ для первых двух этапов.

На трудоемкость алгоритмов часто влияет сама структура представления триангуляции. В [6] подробно описаны различные структуры такого представления и их влияние на трудоемкость. Одна из наиболее распространенных структур триангуляции хранит в себе массив из трех номеров вершин треугольника и массив из трех номеров соседних треугольников. Обход сторон треугольников в такой структуре совершается всегда по часовой стрелке. Такая структура используется и в предлагаемой реализации.

3. Частичная триангуляция в полосах

Когда полосы готовы, в каждой из них методом заметающей прямой строится отдельная невыпуклая триангуляция. Первый треугольник строится из первых двух граничных точек полосы и первой «физической» точки полосы. Далее, продвигаясь по оси X заметающей прямой, берется следующая отсортированная точка. Для нее определяется ближайшее ребро последнего построенного треугольника, строится новый треугольник с вершинами ближайшего ребра и т.д. Кроме того, на этом этапе для каждой полосы создаются списки номеров треугольников, образующих верхнюю и нижнюю границы полосы. Они необходимы для последующего сшивания триангуляций. Трудоемкость данного этапа также имеет порядок $O(n)$.

4. Сшивание триангуляций

Сшивание двух соседних полос производится по граничным ребрам невыпуклых триангуляций. При объединении приходится учитывать множество случаев вырожденности треугольников вследствие невыпуклости границ самих полос. Особенность новой реализации состоит в том, что при объединении полос на их некоторых участках границы полос достраиваются до кусочно-выпуклой оболочки только в особых, нечасто встречающихся случаях.

Идея сшивания сводится к рассмотрению конструкции из трех ребер, как на рис. 1. Здесь S_0, S_1, S_2, S_3 – граничные точки нижней полосы, S_2, S_3, S_4, S_5 – граничные точки верхней полосы. Ведущее ребро t_1b_1 – ребро последнего построенного треугольника при сшивании, t_1t_2 – просматриваемая нижняя граница верхней полосы, а b_1b_2 – соответственно верхняя граница нижней полосы. Во избежание появления вырожденных треугольников необходимо найти такую точку t_n или b_n, чтобы выполнялось условие $t_n > b_2$ или $b_n > t_2$. То есть такую точку верхней полосы, которая по координате X будет лежать дальше, чем текущая точка b_2 нижней полосы. И наоборот, для нижней полосы точка должна лежать по координате X дальше, чем текущая точка t_2 верхней полосы.

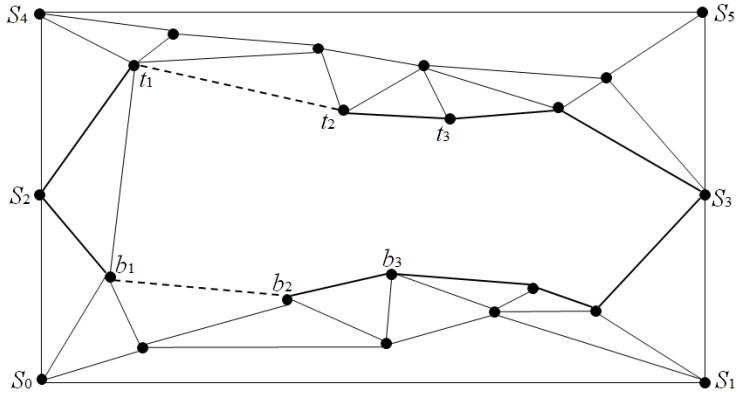

Рис. 1. Пример выбора построения треугольника при сшивании полос. Ситуация $t_3 > b_2$, $b_3 > t_2$
Fig. 1. An example of choosing the construction of a triangle when stripes are merging. Situation $t_3 > b_2$, $b_3 > t_2$

Для простоты опишем алгоритм для модификации нижней границы (для верхней границы – аналогично). Находим точку b_n, лежащую дальше точки t_2 по координате X. Если $n = 3$, то можно построить хотя бы один невырожденный треугольник $t_2b_1t_1$ или $t_1b_2b_1$. Выбор, какой треугольник построить, определяется минимальной длиной нового ведущего ребра t_2b_1 или t_1b_2.

Если же $n > 3$, строится кусочно-выпуклая оболочка на участке границ, содержащих точки $[b_2, b_n]$ (для верхней полосы – аналогично на участке $[t_2, t_n]$). При этом изменяется список номеров треугольников, образующих границу полосы. Данная ситуация проиллюстрирована на рис. 2.

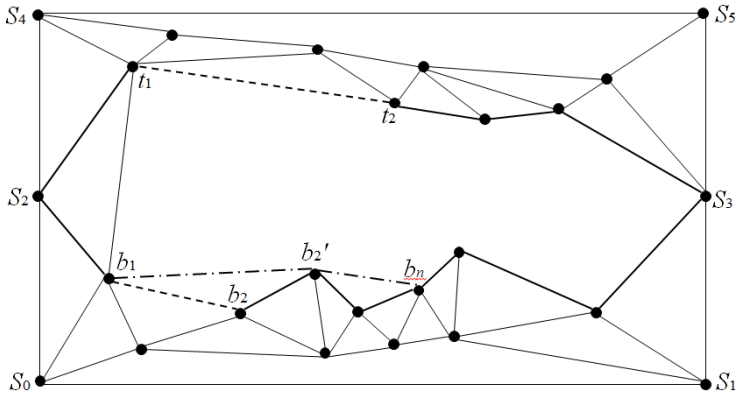

Рис. 2. Достраивание кусочно-выпуклой оболочки при $n > 3$ на участке границ $[b_2, b_n]$.
Новое рассматриваемое ребро при построении треугольников – граница b_1b_2'
Fig. 2. Completion of the piecewise convex hull at $n > 3$ on the boundary segment $[b_2, b_n]$.
Boundary b_1b_2' is a new considered edge at building triangles

Стоит отметить, что первоначально в качестве ведущего ребра t_1b_1 выбирается первое ребро верхней или нижней границы, правая точка которого имеет меньшую координату по оси X. Это вносит единообразие в обработку множества проверок на вырожденность треугольников, в том числе самого первого, при сшивании полос. Тогда следующее ребро выбранной границы становится рассматриваемым, тем самым образуется та же конструкция из ребер t_1t_2, t_1b_1, b_1b_2.

5. Проверка и перестроение треугольников

После сшивания всех полос в единую триангуляцию запускается этап проверки и перестроения треугольников по критерию Делоне. Это позволяет уменьшить количество проверок и перестроений по сравнению с алгоритмом динамического кэширования, в котором перестроение проводится после образования каждого нового треугольника.

На рис. 3 представлены триангуляции до и после этапов сшивания полос и перестроения треугольников на наборе из 5 тыс. точек, полученных при наземном лазерном сканировании участка местности.

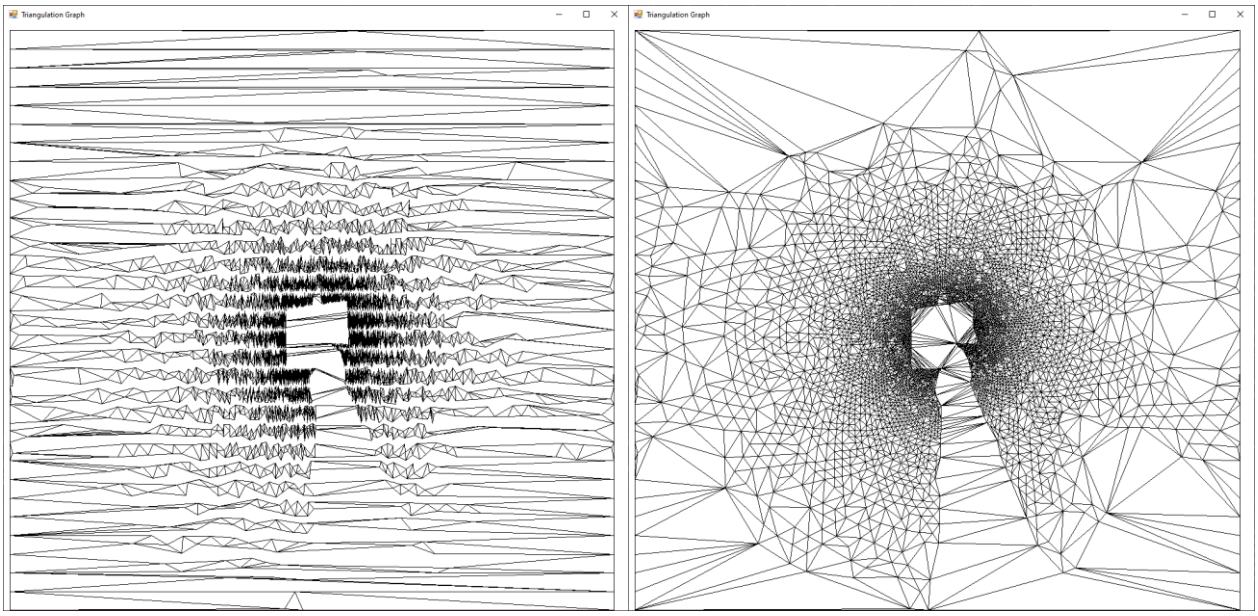

Рис. 3. Полученные триангуляции: до сшивания полос (слева),
после сшивания полос и перестроения треугольников (справа)
Fig. 3. The computed triangulations before stripe merge (on the left), the final merged triangulation (on the right)

6. Сравнение алгоритмов

Опишем вычислительный эксперимент, в ходе которого было проведено качественное сравнение двух алгоритмов: динамического кэширования и невыпуклого полосового слияния. Реализация алгоритмов выполнена на языке C++, для замеров времени используется библиотека <chrono>. Отображения итоговых триангуляций получены программой, написанной на языке C#.

В табл. 1 приведены сравнительные характеристики вычислительного эксперимента для двух алгоритмов триангуляции: DH (Dynamic Hash) – алгоритма с динамическим кэшированием поиска треугольников, SM (Stripe Merging) – алгоритма невыпуклого полосового слияния. Количество точек n, сгенерированных по равномерному распределению, в эксперименте варьирует от 500 тыс. до 4 млн. Качественные характеристики для оценки алгоритмов слева направо: количество точек n, время в секундах t, сумма ребер всех треугольников в триангуляции ($\sum$ ребер), среднее значение минимального угла треугольников (min угол), среднее количество проверок для перестроения треугольников на точку, среднее количество перестроений треугольников на точку. При этом не учитывались треугольники на границах триангуляций, у которых хотя бы одна вершина является граничной точкой. Большинство

таких треугольников очень вытянутые, поэтому они заметно влияют на качественные характеристики итоговой триангуляции. Поэтому в практических случаях эти треугольники, содержащие «искусственные» граничные точки, обычно удаляют.

В алгоритмах использовано два критерия перестроения треугольников – по условию Делоне и по минимальному ребру [5], когда для пары соседних по ребру треугольников, т.е. в четырехугольнике, строится диагональ меньшей длины. При таких проверке и перестроении требуется меньше вычислений, чем по условию Делоне, но полученная триангуляция будет отличаться от триангуляции Делоне. В ряде практических применений может использоваться любая из таких триангуляций, поэтому в табл. 1 приведены характеристики обоих вариантов.

Таблица 1

Качественные характеристики двух алгоритмов построения триангуляции

Алгоритм, критерий	*n*, млн	*t*, с	∑ ребер	min угол, °	Кол-во проверок	Кол-во перестроений
DH, Делоне	0,5	3,99	14 447 526	31,54	20,76	2,79
DH, мин. ребро		3,62	14 286 233	31,93	19,34	2,45
SM, Делоне		2,34	14 649 084	31,56	11,05	1,01
SM, мин. ребро		2,22	14 465 991	32,03	10,75	0,96
DH, Делоне	1	8,11	20 271 128	32,35	20,01	2,61
DH, мин. ребро		7,46	20 097 236	32,73	18,69	2,3
SM, Делоне		4,67	20 671 102	32,37	10,35	0,91
SM, мин. ребро		4,33	20 465 094	32,82	10,06	0,85
DH, Делоне	2	16,41	28 096 636	33,78	18,59	2,31
DH, мин. ребро		14,97	27 936 010	34,17	17,51	2,06
SM, Делоне		8,81	28 970 376	33,82	9,33	0,77
SM, мин. ребро		8,49	28 773 724	34,24	9,16	0,73
DH, Делоне	4	29,21	38 158 768	36,35	16,18	1,87
DH, мин. ребро		26,91	37 996 544	36,75	15,45	1,7
SM, Делоне		15,51	39 799 132	36,41	7,75	0,56
SM, мин. ребро		15,14	39 617 344	36,82	7,7	0,55

Проанализируем работу алгоритмов на примере набора из 500 тыс. точек. Как видно из табл. 1, предлагаемый алгоритм SM отрабатывает быстрее алгоритма DH в 1,7 раза. При этом отличия триангуляций, полученные этими алгоритмами, несущественны и объясняются тем, что в алгоритме DH добавляется всего 4 дополнительных граничных точки, а в алгоритме SM – по 4 точки в каждой полосе. Среднее количество проверок по критерию Делоне на одну точку в алгоритме SM почти в 2 раза меньше, чем в алгоритме DH, а количество перестроений почти в 3 раза меньше.

Если провести анализ по различиям полученных результатов по критериям перестроения Делоне и минимального ребра, то можно увидеть, что алгоритм с перестроением по условию минимального ребра работает чуть быстрее и требует чуть меньше проверок. Интересно, что среднее значение минимальных углов треугольников у этого алгоритма больше, т.е. в совокупности все треугольники оказались ближе к равносторонним.

Таблица 2

Распределение времени на этапах полосового алгоритма

Распределение времени	0,5 млн	% от *t*	1 млн	% от *t*
Разбиение пространства и точек на полосы	0,16	7,33	0,31	7,35
Общее время сортировки точек во всех полосах	0,31	14,23	0,69	16,51
Общее время частичной триангуляции по полосам	0,37	16,83	0,68	16,26
Общее время объединения всех полос	0,34	15,55	0,67	15,91
Проверка и перестроение всех треугольников	1,01	46,06	1,84	43,96
Общее время выполнения алгоритма, *t*	2,19	–	4,19	–

Так как предлагаемый алгоритм состоит из нескольких этапов, то интересно отследить вклад каждого этапа в общее время работы. Рассмотрим время выполнения основных этапов на двух примерах: из 500 тыс. и 1 млн точек. Как видно из табл. 2, наибольшее влияние на общее время алгоритма имеет этап проверки и перестроения треугольников, его время составляет примерно 45% от общего времени. Этапы сортировки точек в полосах, построения триангуляции в полосах, объединения полос имеют примерно одинаковый вклад в процентном соотношении.

7. Поведение алгоритма при неравномерном распределении точек

Также было исследовано, как поведет себя алгоритм при построении триангуляции на точках, сгенерированных по неравномерному распределению. Моделью такого распределения внутри области, например квадрата, может служить фрактальное распределение с точкой сгущения в центре квадрата.

Один из вариантов одномерного фрактального распределения в диапазоне [0, 1] задается следующей интегральной функцией распределения [12]:

$$\begin{cases} y = x^{1/r},\ 0 \le x \le 1, \\ y = 0,\ x \le 0, \\ y = 1,\ x \ge 1, \end{cases} \tag{3}$$

где $r \ge 1$ – ранг распределения.

Если $r = 1$, то это распределение превращается в равномерное. Чем больше r, тем больше распределение отличается от равномерного. Это распределение самоподобно, так как если взять часть кривой в диапазоне $[0, x_0]$, умножить все $x \le x_0$ на $1/x_0$, то соответствующие значения функции совпадут с исходными на всем диапазоне [0, 1].

При генерации координат случайных точек по этому распределению нужно датчиком случайных чисел сгенерировать две независимые случайные величины d_0 и d_1 с равномерным распределением в диапазоне [0, 1). Затем вычислить случайное расстояние от центра квадрата $D \times D$ через функцию, обратную к функции (3):

$$R = d_0^{\,r}\, D/\sqrt{2}\ . \tag{4}$$

Тогда координаты точки относительно центра квадрата вычисляются по формулам

$$x = R\cos(2\pi d_1),\ \ y = R\sin(2\pi d_1), \tag{5}$$

причем если точка выходит за пределы квадрата, то она отбрасывается.

В табл. 3 приведены качественные характеристики работы двух алгоритмов для фрактального распределения при $r = 1{,}5$ на наборах из 500 тыс. и 1 млн точек.

Таблица 3

Характеристики алгоритмов при обработке точек фрактального распределения, r = 1,5

Алгоритм, критерий	n, млн	t, с	∑ ребер	min угол, °	Кол-во проверок	Кол-во перестроений
DH, Делоне	0,5	4,27	12 237 120	32,85	17,78	2,32
DH, мин. ребро		3,91	12 102 218	33,23	16,65	2,04
SM, Делоне		2,42	12 281 369	32,87	10,83	1,13
SM, мин. ребро		2,44	12 124 610	33,31	10,68	1,10
DH, Делоне	1	8,24	17 010 980	33,73	16,41	2,08
DH, мин. ребро		7,56	16 853 486	34,09	15,41	1,84
SM, Делоне		5,1	17 109 162	33,75	9,51	0,93
SM, мин. ребро		4,72	16 925 260	34,16	9,39	0,91

Как и ожидалось, время работы обоих алгоритмов увеличилось, но всего на 5%. Другие характеристики построенных триангуляций изменились несущественно. Таким образом, предложенный алго-

ритм работает достаточно эффективно как для равномерных, так и для неравномерных (фрактальных) распределений.

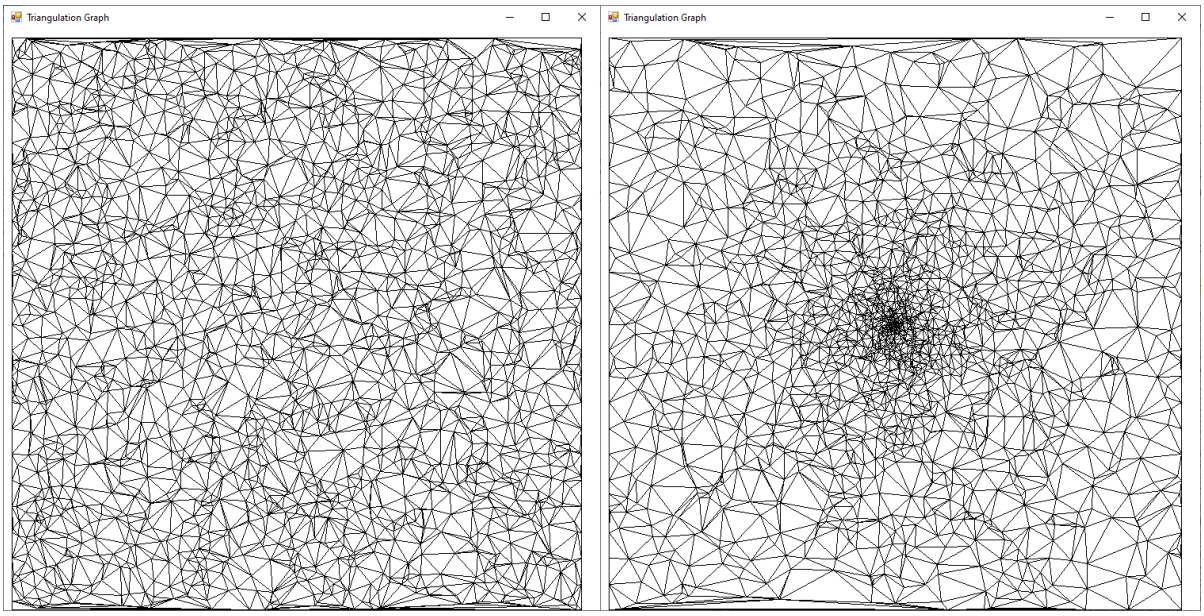

Рис. 4. Триангуляция по набору из 2 000 точек с равномерным распределением (слева) и с неравномерным (фрактальным, r = 1,5) распределением (справа)
Fig. 4. Delaunay triangulation of a set of 2000 points with uniform distribution (on the left) and with non-uniform (fractal, r = 1.5) distribution (on the right)

На рис. 4 приведен пример построенной триангуляции предложенным полосовым алгоритмом по набору из 2 000 точек для равномерного и неравномерного (фрактального) распределения.

Заключение

В данной статье описана быстрая реализация полосового алгоритма триангуляции Делоне, время работы которого в 1,7 раза меньше по сравнению с известным наиболее быстрым алгоритмом динамического кэширования поиска треугольников. Этого удалось добиться за счет следующих модификаций:

1) к каждой полосе точек добавлено 4 граничные точки; кроме того, отсортированные внутри полосы точки по оси X при одинаковых координатах X отсортированы по Y, что гарантирует отсутствие вырожденных треугольников при триангуляции внутри полосы;

2) в процессе сшивания полос при выборе очередного сшивающего ребра вначале выполняются самые простые проверки расположения точек на границах полос, гарантирующие построение невырожденных треугольников, и только если эти проверки безуспешны, выполняются более сложные проверки и перестроение участка одной из границ до частичной выпуклости.

Благодаря этим модификациям количество проверок треугольников уменьшилось почти в 2 раза, а количество перестроений треугольников – почти в 3 раза по сравнению с алгоритмом динамического кэширования. Как показал вычислительный эксперимент, эти выводы справедливы для области точек как с равномерным, так и неравномерным (фрактальным) распределением.

Кроме того, в вычислительном эксперименте проверялась работа алгоритмов с перестроением треугольников не только по критерию Делоне, но и по минимальному ребру, когда получается триангуляция, вполне пригодная для практического применения. При этом время работы алгоритмов сокращается на 3–5%, а такие общие характеристики триангуляции, как сумма длин всех ребер всех треугольников и среднее значение минимальных углов во всех треугольниках, изменяется незначительно.

Преимуществом предложенного алгоритма является не только его более высокая скорость работы, но и возможность распараллеливания на десятки или даже сотни потоков на основных этапах выполнения.

Список источников

1. Костюк Ю.Л., Гульбин К.Г., Пешехонов С.В. Построение поверхностной триангуляции и выделение пространственных фигур по данным лазерного сканирования // Вестник Томского государственного университета. Сер. «Информатика, кибернетика, математика». 2006. № 293. С. 151–155.
2. Костюк Ю.Л., Литовченко М.И. Распознавание граней трехмерных объектов по данным лазерного сканирования // Информационные технологии и математическое моделирование (ИТММ–2017) : материалы XVI Междунар. конф. им. А.Ф. Терпугова. Томск : Изд-во НТЛ, 2017. С. 55–61.
3. Храмов В.В. Способ описания объектов сцены системы технического зрения робота средствами нечеткой триангуляции Делоне с использованием адаптивной сетки // Современные информационные технологии и ИТ-образование. 2020. Т. 16, № 4. С. 833–840. doi: 10.25559/SITITO.16.202004.833-840
4. Крамаров С.О., Митясова О.Ю., Темкин И.О., Храмов В.В. Методология интеллектуальной навигации для управления автономными подвижными объектами на основе триангуляции Делоне // Горный информационно-аналитический бюллетень. 2021. № 2. С. 87–98. doi: 10.25018/0236-1493-2021-2-0-87-98
5. Препарата Ф., Шеймос М. Вычислительная геометрия: введение : пер. с англ. М. : Мир, 1989.
6. Скворцов А.В. Триангуляция Делоне и ее применение. Томск : Изд-во Том. ун-та, 2002.
7. Elshakhs Yu.S. et al. A comprehensive survey on Delaunay triangulation: applications, algorithms, and implementations over CPUs, GPUs, and FPGAs // IEEE Access. 2024. V. 12. P. 12562–12585. doi: 10.1109/ACCESS.2024.3354709
8. Hadi N.A., Farhani A., Dahalan W.M. An Improved Simple Sweep Line Algorithm for Delaunay Refinement Triangulation // Advanced Engineering for Processes and Technologies II / A. Ismail, W.M. Dahalan, A. Öchsner (eds.). Springer, 2021. С. 263–270. doi: 10.1007/978-3-030-67307-9_23
9. Dimitriou P., Karyotis V. On the computation of Delaunay triangulations via genetic algorithms // Evolutionary Intelligence. 2024. V. 17 (4). P. 2413–2432. doi:10.1007/s12065-023-00893-5
10. Бикбулатов Т.Х., Тумаков Д.Н. Использование частичной параллелизации для триангуляции двумерных областей // Программные продукты и системы. 2022. Т. 35, № 3. С. 293–304. doi: 10.15827/0236-235X.139.293-304
11. Скворцов А.В., Костюк Ю.Л. Эффективные алгоритмы построения триангуляции Делоне // Геоинформатика. Теория и практика. 1998. Вып. 1. С. 22–47.
12. Костюк Ю.Л. Эффективные алгоритмы обработки и отображения графических данных и их реализация в программных комплексах : автореф. … дис. д-ра техн. наук. Томск, 2002. URL: http://vital.lib.tsu.ru/vital/access/manager/Repository/vtls:000139431 (дата обращения: 26.12.2024).

References

1. Kostyuk, Yu.L., Gulbin, K.G. & Peshekhonov, S.V. (2006) Postroenie poverkhnostnoy triangulyatsii i vydelenie prostranstvennykh figur po dannym lazernogo skanirovaniya [Construction of surface triangulation and selection of spatial figures based on laser scanning data]. *Vestnik Tomskogo gosudarstvennogo universiteta. Ser. "Informatika, kibernetika, matematika."* 293. pp. 151–155.
2. Kostyuk, Yu.L. & Litovchenko, M.I. (2017) Raspoznavanie graney trekhmernykh ob"ektov po dannym lazernogo skanirovaniya [Recognition of edges of three-dimensional objects based on laser scanning data]. *Informatsionnye tekhnologii i matematicheskoe modelirovanie (ITMM–2017)* [Information Technologies and Mathematical Modeling (ITMM-2017)]. Proc. of the 16th International Conference named after A.F. Terpugov. Tomsk: NTL. pp. 55–61.
3. Khramov, V.V. (2020) Sposob opisaniya ob"ektov stseny sistemy tekhnicheskogo zreniya robota sredstvami nechetkoy triangulyatsii Delone s ispol'zovaniem adaptivnoy setki [A Method of Describing Objects in a Scene of a Robot Vision System by Means of Fuzzy Delaunay Triangulation Using an Adaptive Mesh]. *Sovremennye informatsionnye tekhnologii i IT-obrazovanie*. 16(4). pp. 833–840. DOI: 10.25559/SITITO.16.202004.833-840
4. Kramarov, S.O., Mityasova, O.Yu., Temkin, I.O. & Khramov, V.V. (2021) Metodologiya intellektual'noy navigatsii dlya upravleniya avtonomnymi podvizhnymi ob"ektami na osnove triangulyatsii Delone [Delaunay triangulation-based methodology of intelligent navigation and control of mobile objects]. *Gornyy informatsionno-analiticheskiy byulleten'*. 2. pp. 87–98. DOI: 10.25018/0236-1493-2021-2-0-87-98
5. Preparata, F.P. & Shamos, M.I. (1989) *Vychislitel'naya geometriya: vvedenie* [Computational Geometry: An Introduction]. Translated from English). Moscow: Mir.
6. Skvortsov, A.V. (2002) *Triangulyatsiya Delone i ee primenenie* [Delaunay Triangulation and its Application]. Tomsk: Tomsk State University.
7. Elshakhs, Yu.S. et al. (2024) A comprehensive survey on Delaunay triangulation: applications, algorithms, and implementations over CPUs, GPUs, and FPGAs. *IEEE Access*. 12. DOI: 10.1109/ACCESS.2024.3354709
8. Hadi, N.A., Farhani, A. & Dahalan, W.M. (2021) An improved simple sweep line algorithm for delaunay refinement triangulation. In: Ismail, A., Dahalan, W.M. & Öchsner, A. (eds) *Advanced Engineering for Processes and Technologies II*. Springer. pp. 263–270. DOI: 10.1007/978-3-030-67307-9_23
9. Dimitriou, P. & Karyotis, V. (2024) On the computation of Delaunay triangulations via genetic algorithms. *Evolutionary Intelligence*. 17(4). pp. 2413–2432. DOI: 10.1007/s12065-023-00893-5

10. Bikbulatov, T.Kh. & Tumakov, D.N. (2022) Ispol'zovanie chastichnoy parallelizatsii dlya triangulyatsii dvumernykh oblastey [Using partial parallelization to triangulate 2D domains]. *Programmnye produkty i sistemy – Software & Systems*. 35(3). pp. 293–304. DOI: 10.15827/0236-235X.139.293-304
11. Skvortsov, A.V. & Kostyuk, Yu.L. (1998) Effektivnye algoritmy postroeniya triangulyatsii Delone [Efficient algorithms for constructing Delaunay triangulation]. *Geoinformatika. Teoriya i praktika – Geoinformatics. Theory and Practice*. 1. pp. 22–47.
12. Kostyuk, Yu.L. (2002) *Effektivnye algoritmy obrabotki i otobrazheniya graficheskikh dannykh i ikh realizatsiya v programmnykh kompleksakh* [Effective algorithms for processing and displaying graphic data and their implementation in software systems]. Abstract of Engineering Dr. Diss. Tomsk: Tomsk State University.

***Информация об авторах*:**

Литовченко Марина Игоревна – ассистент, аспирант кафедры теоретических основ информатики Института прикладной математики и компьютерных наук Национального исследовательского Томского государственного университета (Томск, Россия). E-mail: litovchenkom21@yandex.ru

Костюк Юрий Леонидович – профессор, доктор технических наук, профессор кафедры теоретических основ информатики Института прикладной математики и компьютерных наук Национального исследовательского Томского государственного университета (Томск, Россия). E-mail: kostyuk.yu48@mail.ru

Вклад авторов: все авторы сделали эквивалентный вклад в подготовку публикации. Авторы заявляют об отсутствии конфликта интересов.

***Information about the authors*:**

Litovchenko Marina I. (Assistant, Post-graduate Student, National Research Tomsk State University, Tomsk, Russian Federation). E-mail: litovchenkom21@yandex.ru

Kostyuk Yuriy L. (Professor, Doctor of Technical Sciences, National Research Tomsk State University, Tomsk, Russian Federation). E-mail: kostyuk.yu48@mail.ru

Contribution of the authors: the authors contributed equally to this article. The authors declare no conflicts of interests.

Поступила в редакцию 16.01.2025; принята к публикации 02.06.2025

Received 16.01.2025; accepted for publication 02.06.2025