The notion \( k \)-th order Voronoi diagram of a finite set of points in the Euclidean plane \( \mathbb{E}^2 \) is generalized to the \( k \)-nearest-neighbor Voronoi diagram of a finite set of convex polygons, line segments and points and they are characterized by some interesting theorems.

Furthermore given \( n \) convex polygons with a maximum number of \( m \) vertices and a total number of \( M \) vertices, we present an algorithm for constructing simultaneously all \( k \)-nearest-neighbor Voronoi diagrams, \( k \in \{1, \ldots, n-1\} \), that takes \( O(n^2(n+m)M) \) time and \( O(n^2(n^2+M)) \) space.

We can also apply that algorithm efficiently to a CREW-PRAM with \( 2^n \) processors, where it runs in \( O(m(n+m)) \) time and \( O(n^2(n^2+M)) \) space.

The algorithm is also shown to be applicable under more general convex objects if certain conditions are satisfied.

There are several applications in motion planning, pattern recognition, clustering algorithms, etc.

1 Introduction

One of the most fundamental problems in computational geometry is the \( k \)-nearest-neighbor problem, a variant of the classical nearest-neighbor problem.

The problem is to find, among a set \( S \) of \( n \) objects in a space \( E \), the \( k \) nearest objects to a given test point \( q \in E \) with regard to a general distance measure \( d \).

To solve the \( k \)-nearest-neighbor problem for a finite set of points in the Euclidean plane \( \mathbb{E}^2 \), Shamos and Hoey [ShHo 75] proposed an approach using Voronoi diagrams. They introduced the \( k \)-nearest-neighbor Voronoi diagram \( V_k(S) \), that subdivides \( \mathbb{E}^2 \) into maximal regions, so that all points within a given region have the same \( k \) nearest neighbors.
The first algorithm for computing $V_k(S)$ was presented by Lee [Le 82]; this method required $O(k^2 n \log n)$ time and $O(k^2 (n - k))$ space. Later Edelsbrunner [Ed 86] reported an other technique to construct the diagram in $O(k (n - k) \sqrt{n} \log n)$ time and optimal $O(k (n - k))$ storage. Shortly afterwards Chazelle and Edelsbrunner [ChEd 87] presented two versions of an algorithm for constructing $V_k(S)$: the first one requires $O(n^2 \log n + k (n - k) \log^2 n)$ time and optimal $O(k (n - k))$ storage, while with additional $O(n^2)$ preprocessing and storage, the other version speeds up the computation to $O(n^2 + k (n - k) \log^2 n)$.

On the other hand Dehne [De 83] evolved an algorithm for constructing all Voronoi diagrams $V_1(S), \ldots, V_{n-1}(S)$ in $O(n^4)$ time and space, which was later improved by an algorithm of Edelsbrunner and Seidel [EdSe 86] that takes only optimal $O(n^3)$ time and storage.

The main result of the present work consists in the generalization of the underlying objects. We present $k$-nearest-neighbor Voronoi diagrams of convex polygons, line segments and points in the Euclidean plane $\mathbb{E}^2$. After a detailed examination of the bisector of two convex polygons in the second section and the Voronoi diagrams in the third section we develop an algorithm for constructing simultaneously all Voronoi diagrams $V_1(S), \ldots, V_{n-1}(S)$ in $O(n^2 (n + m) M)$ time and $O(n^2 (n^2 + M))$ storage in section four. At last we apply that algorithm efficiently to a CREW-PRAM with $2 \binom{n}{3}$ processors where it takes $O(m(n + m))$ time and $O(n^2(n^2 + M))$ space.

These new Voronoi diagrams are also a generalization of first order Voronoi diagrams of a set of line segments which were introduced by Drysdale and Lee [DrLe 78] and extensively studied in the last years by Fortune [Fo 86] and Yap [Ya 87].

2 The Bisector of two Convex Polygons

Given a finite set of convex polygons

$$S := \{Pol_1, \ldots, Pol_n\}$$

in the Euclidean plane $\mathbb{E}^2$, $|S| = n \geq 3$, under the two following assumptions:

A The polygons in $S$ are disjoint.

B There is no point in the Euclidean plane with the same distance to four different polygons in $S$.

Let every polygon $Pol_i \in S$ be represented by a cyclically ordered sequence $P_{i1}, \ldots, P_{il(i)}$ of its vertices.\(^2\)

--

\(^1\)A comparison between these four algorithms is drawn in [ChEd 87].

\(^2\)l(i) denotes the number of vertices of $Pol_i \in S$. If $Pol_i$ degenerates into a line segment or a point, then $l(i) = 2$ and $l(i) = 1$ respectively.