A Hybrid 2-D Delaunay Triangulation Algorithm Exploiting the GPU Parallelism

Petros Vasileiou and Emmanouil Psarakis

Dept. of Computer Engineering and Informatics, University of Patras, Greece

Delaunay Triangulation Problem

Let $T(P)$ be a triangulation of a given 2-D point set P having $3|T(P)|$ angles contained in set $A(T(P)) = \{a_i,\ i= 1,2,\ldots,3|T(P)|\}$. Let $L_{A(T(P))}$ be the sorted, in ascending order, angle list of $T(P)$ triangulation. Then the Delaunay Triangulation $T(P)$ can result from the solution of the following optimization problem:

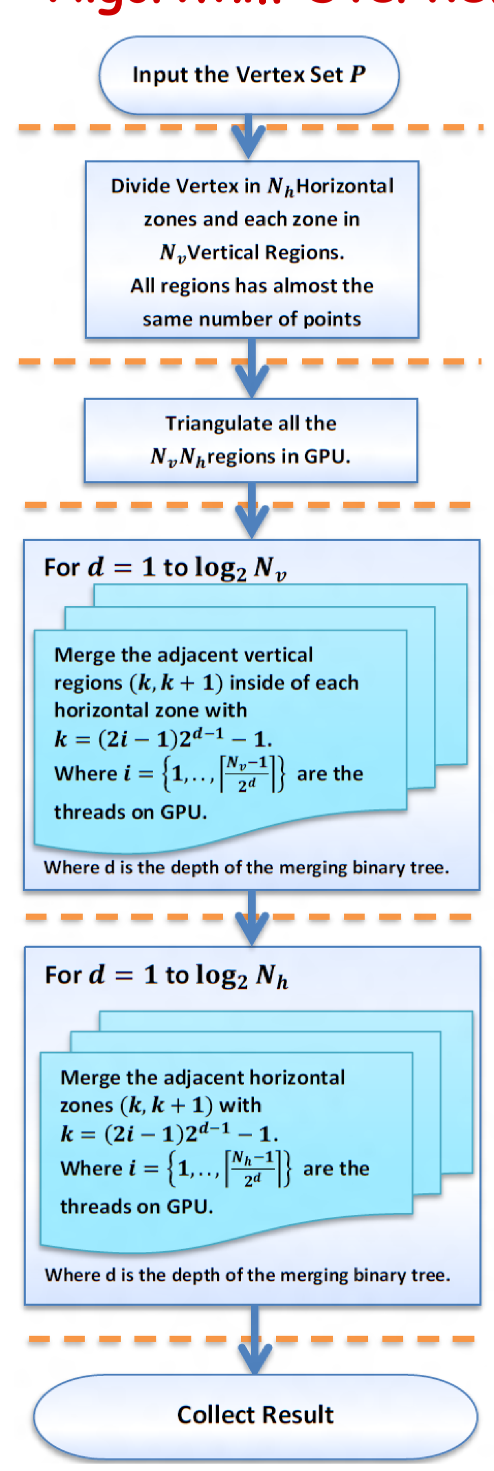
$$T_D(P) = lex \inf_{T(P)} L_{A(T(P))}$$

i.e. DT constitutes the lexicographic infimum over all angle lists formed from the triangulations of the given point-set P.

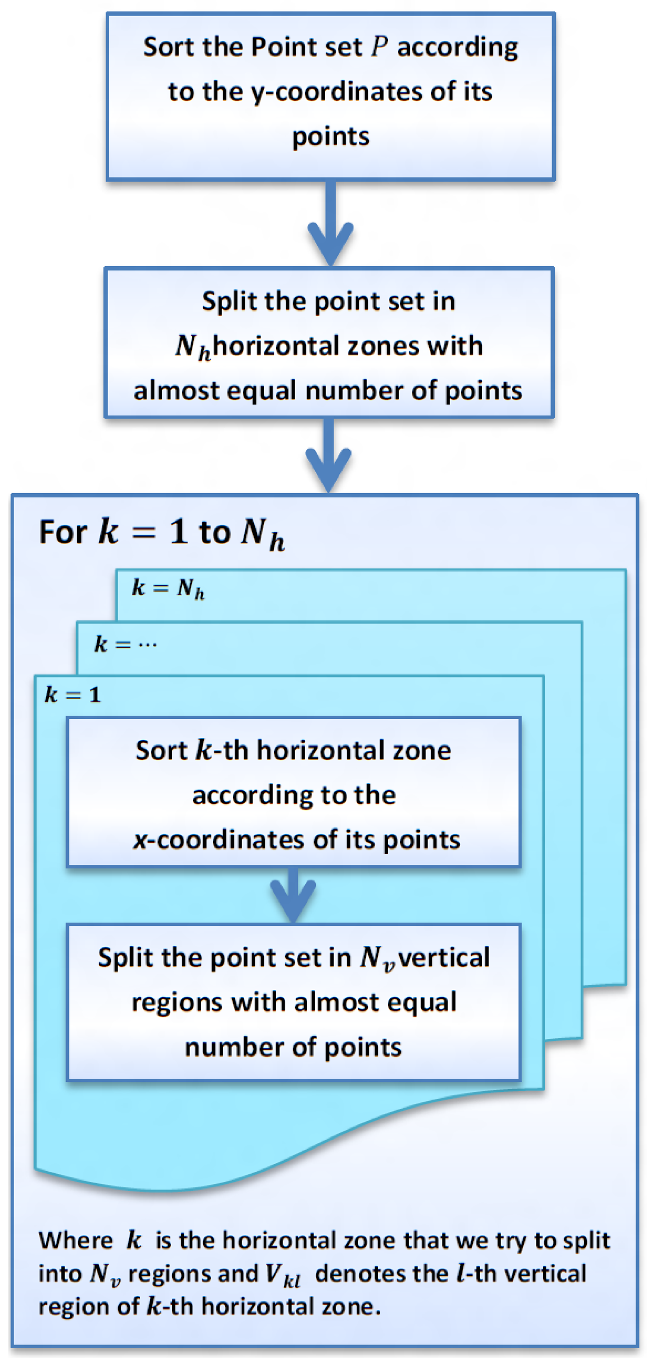
Characteristics of our approach

- Exploits the high parallelism offered by GPU while simultaneously runs exclusively on GPU under the guidance of CPU, but without the need of any feedback from GPU.
- Enjoys performance which is almost independent from the distribution of the point-set.
- Reduces the necessary memory footprint by using the classical geometric data structure "half-edges", which is widely used in CPU-based implementations of many computational geometry algorithms.

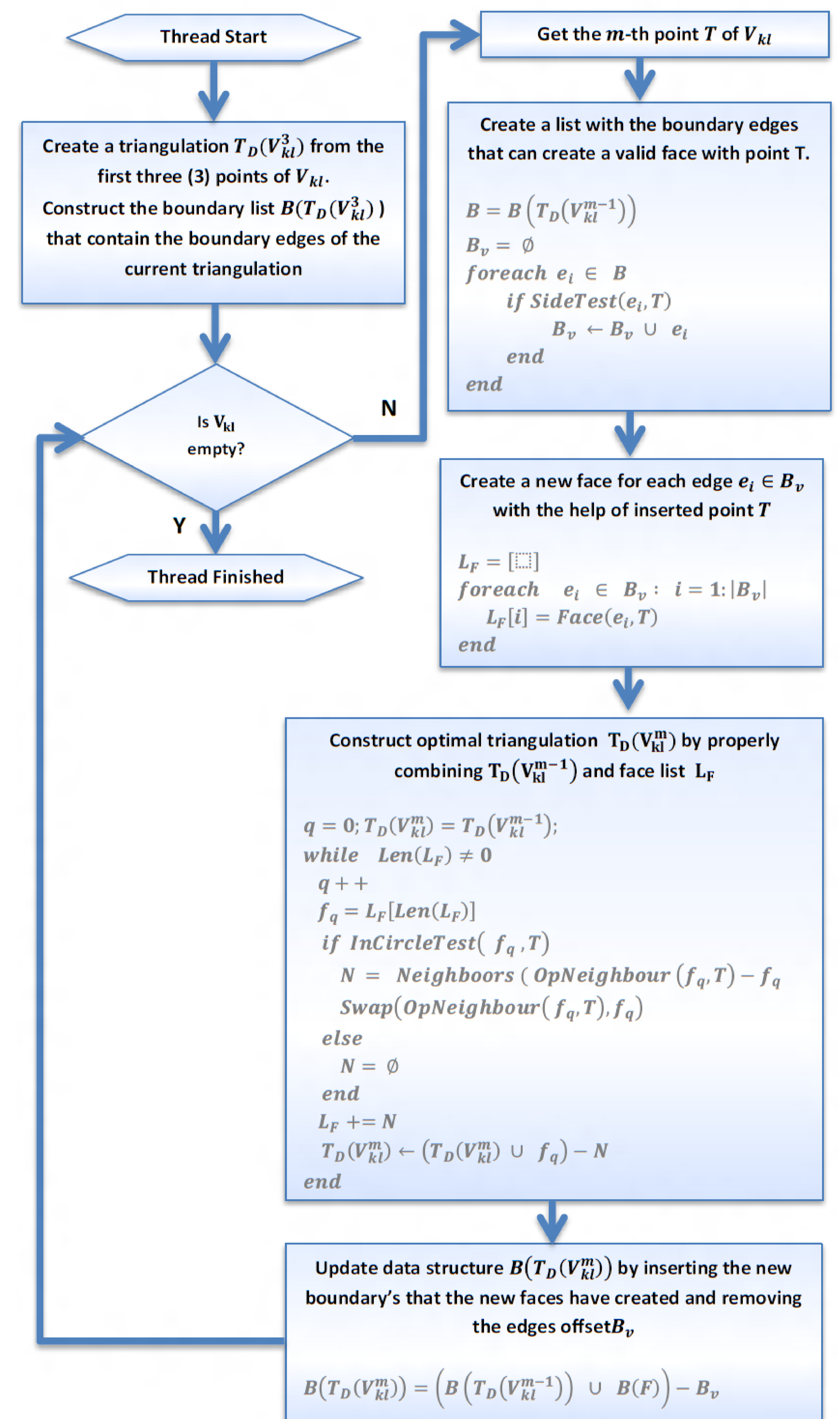
Algorithm Overview

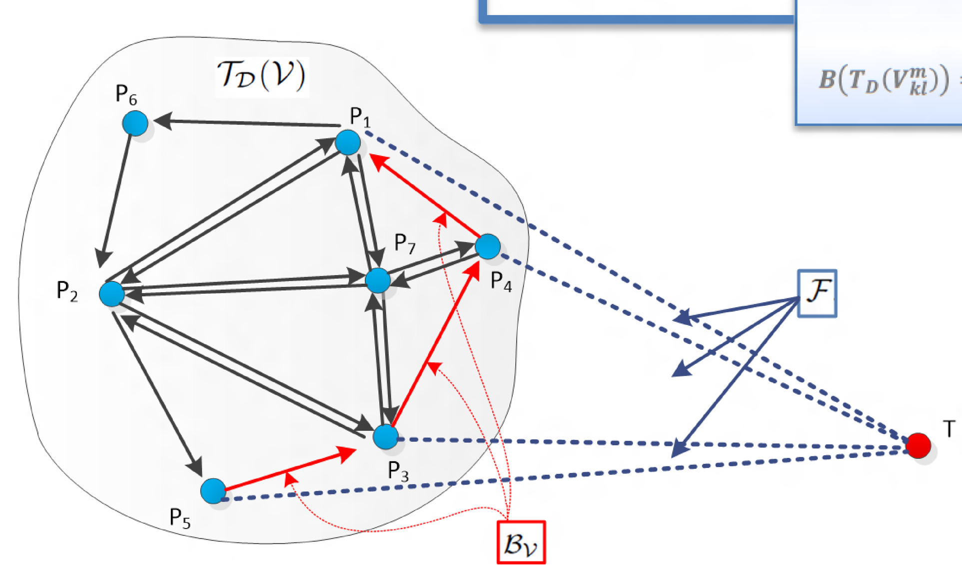


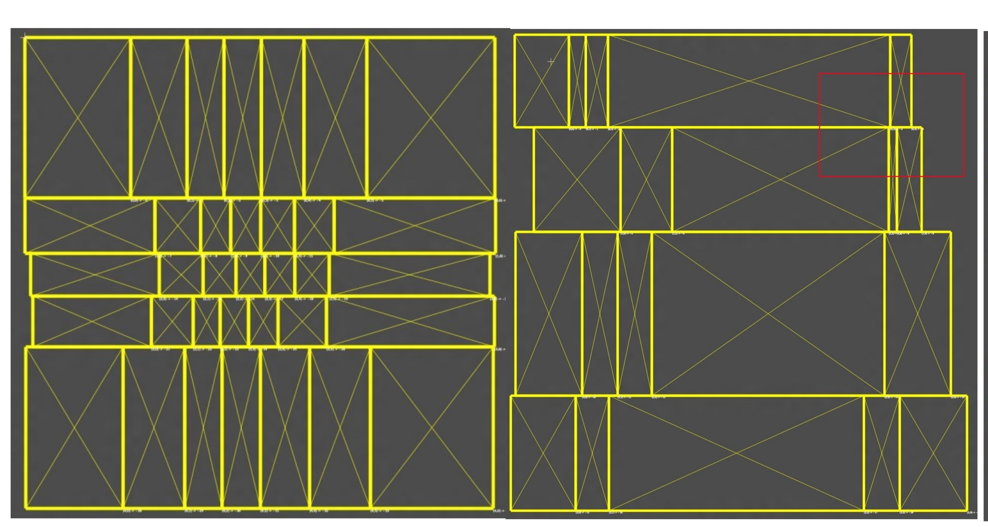
S_1 : Partitioning of the given 2-D point-set P in a specified number of subsets. Each subset is sorted according to the x-coordinates of its points.



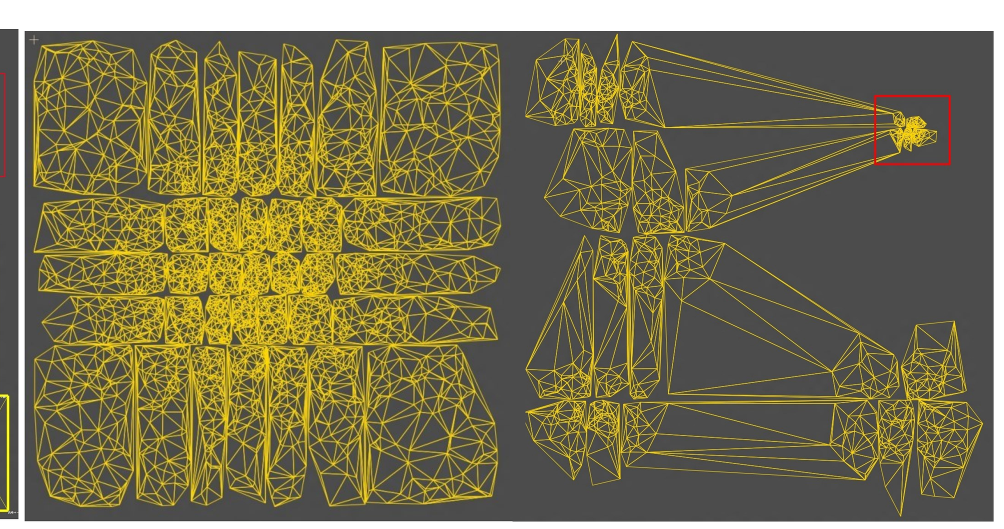
S_2 : Delaunay Triangulation of each subset by using an *order recursive* insertion algorithm tailored to exploit both GPU architecture and the special form of the sequence of the insertion problems imposed by Step S_1 of the algorithm.







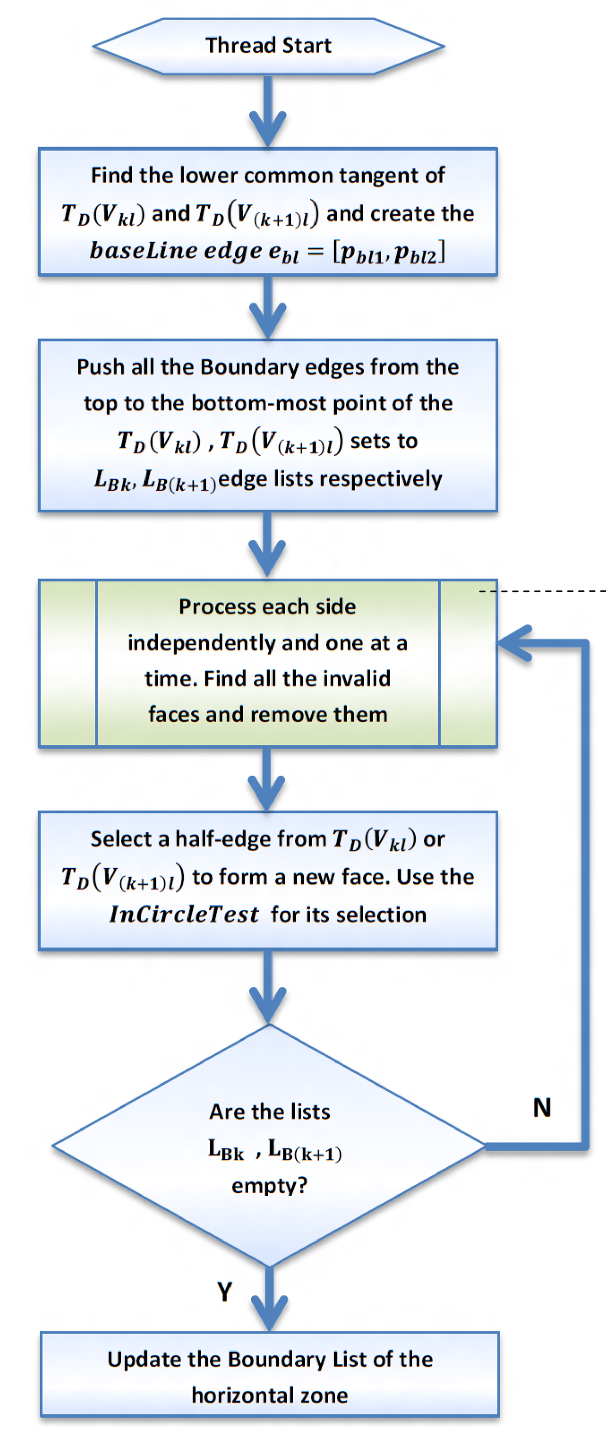
Partitioning obtained from the application of the proposed algorithm (step S_1) on a Gaussian and a mixture of Gaussian distributed point sets.

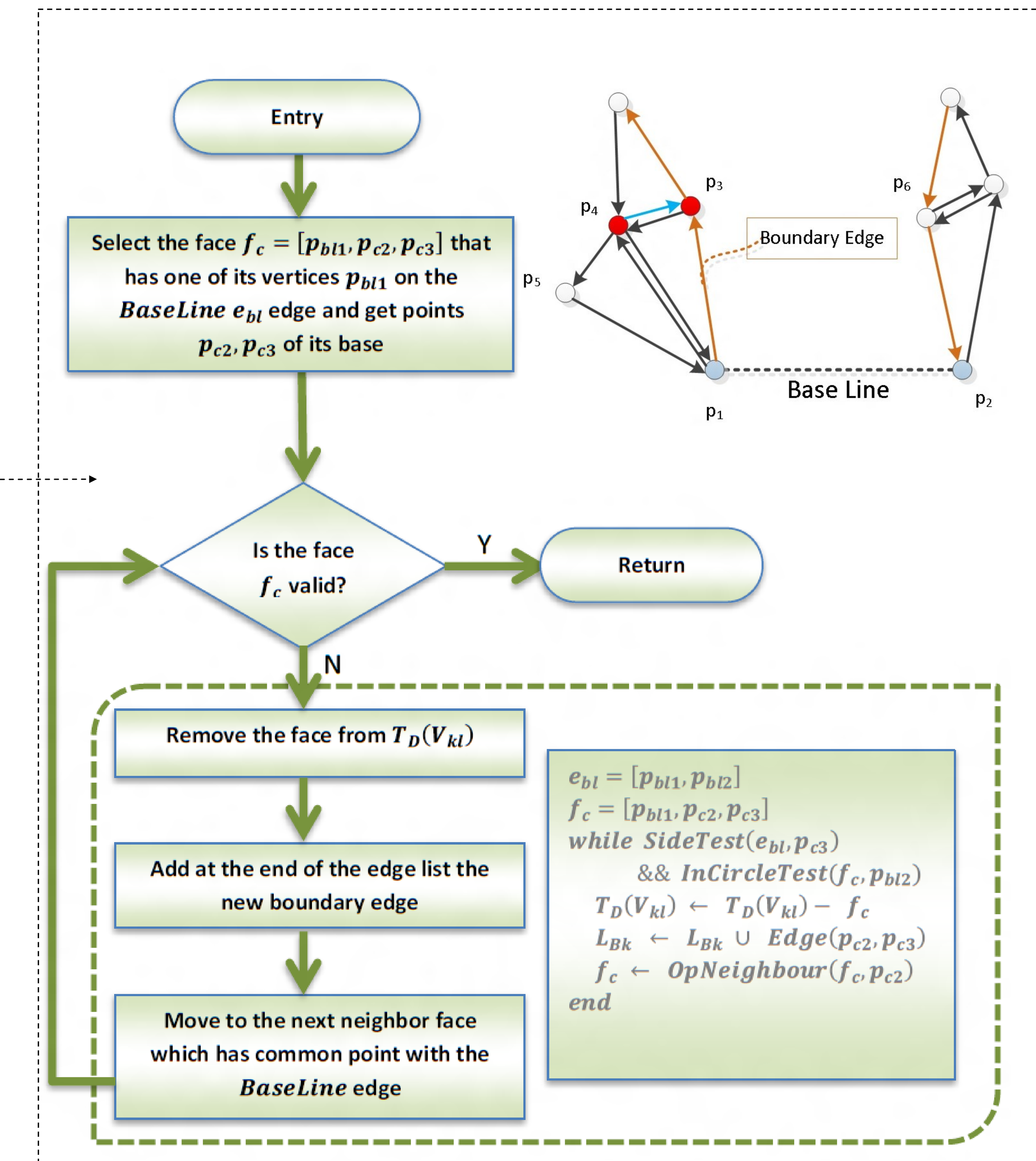


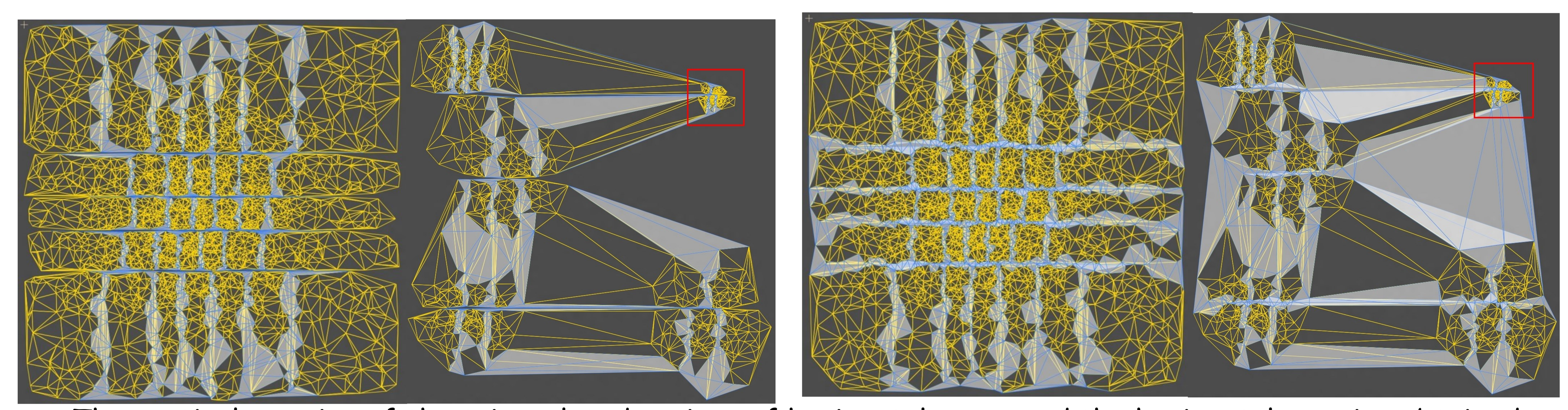
The Delaunay Triangulation T_D of the V_{kl} subsets after the application of step S_2 of the proposed algorithm.

S_3 : Merging of the vertical subsets in each horizontal zone. This step is implemented by mapping all the vertical subsets of each horizontal zone onto a corresponding binary tree whose depth specifies the complexity of this specific step. As it is clear, this complexity is an increasing function of the depth of binary tree which, in turn, is a decreasing function of the cardinality of subsets.

S_4 : Merging of the horizontal zones. This step is similar to the previous one, but it is, instead, applied onto the horizontal zones.







The vertical merging of the triangulated regions of horizontal zones and the horizontal merging obtained from the application of step S_3 and S_4 of the proposed algorithm.

Experimental Results

We have applied the proposed algorithm, and its rivals, in a number of uniformly and non-uniformly distributed point sets with their cardinalities ranging from 50K up to 1M points. The performance of the proposed algorithm in terms of running time needed for the formation of the optimum set $T_D(P)$, clearly outperforms CPU-based, as well as, state of the art GPU-based implementations of DT algorithms.