Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2015, Article ID 720249, 7 pages
http://dx.doi.org/10.1155/2015/720249

Research Article

A Novel Model of Conforming Delaunay Triangulation for Sensor Network Configuration

Yan Ma, Yan-ling Hao, and Feng-min Tian

College of Automation, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Yan Ma; mayan0443@hrbeu.edu.cn

Received 19 August 2014; Accepted 10 September 2014

Academic Editor: Zheng-Guang Wu

Delaunay refinement is a technique for generating unstructured meshes of triangles for sensor network configuration engineering practice. A new method for solving Delaunay triangulation problem is proposed in this paper, which is called endpoint triangle's circumcircle model (ETCM). As compared with the original fractional node refinement algorithms, the proposed algorithm can get well refinement stability with least time cost. Simulations are performed under five aspects including refinement stability, the number of additional nodes, time cost, mesh quality after intruding additional nodes, and the aspect ratio improved by single additional node. All experimental results show the advantages of the proposed algorithm as compared with the existing algorithms and confirm the algorithm analysis sufficiently.

1. Introduction

Recently, the concept of intelligent network system is very popular in the world. Actually, how can we deploy and optimize the sensor? It is still a difficult issue to scientists that affects both cost and detection capability, which are required considerations of both coverage and connectivity. A sensor node may perform the dual function of sensing the environment and acting as a relay node. In a real sensor network system, all sensor nodes distribute as a discrete data set, which will form a mesh network to provide monitoring of the environment. The terms mesh network will be used throughout this paper to describe a sensor network configuration [1].

Delaunay triangulation (DT) is an effective method to carve up a discrete data region, which is especially widely used in sensor network configuration engineering field [2–5]. In most cases, there is a constricting relationship among the discrete data. The discrete data may comprise some vector lines and close polygons, which must be included in the result of partition. In general, the Delaunay triangulation will not contain all edges of the graph. So far there are two types of DT algorithm: constrained Delaunay triangulation [6, 7] and conforming Delaunay triangulation [8, 9]. The former method is a best approximation of the Delaunay triangulation, given that it must contain all features in the graph. Generally, the DT property cannot be preserved and the quality of the mesh declines in constrained Delaunay triangulation, which will influence the stability and convergence of finite element numerical calculation. In the meanwhile, the conforming DT method can be considered as a degenerate Delaunay triangulation, whose relationships to the constrained graph are that each vertex of the graph is a vertex of the triangulation and each edge of the graph is a union of edges of the triangulation and also satisfies the DT property. But any introduced new node will lead the original graph to change. Therefore, the method should be used restrainedly depending on the actual situation.

Usually, constructing conforming DT is more difficult than constructing constrained DT, as it requires a number of points to achieve conformity. The core technique of the conforming DT is to subdivide the constraints. This paper presents a novel node refinement algorithm, which has better triangulation quality and fewer additional nodes than other algorithms.

The rest of this paper is organized as follows. In Section 2, the basic Delaunay triangulation problem is briefly introduced and the main idea of endpoint triangle's circumcircle model (ECTM) is described in detail in Section 3. Then the convergence and complexity of ECTM are analyzed in Sections 4 and 5. Section 6 gives the simulation experiments' results of the new ECTM with other methods. Finally, a conclusion is drawn in Section 7.

2. Problem Description

Suppose $\Omega(P, B)$ is a planer straight line graph, where $B = B_0, B_1, \ldots, B_M$, B_0 is exterior feature constraint, $B_j > 0$ are interior feature constraints, and $P = P_1, P_2, \ldots, P_N$ is the collection of all discrete points and endpoints of feature lines.

If B_i is a close constraint, r_i describes the single connected domain of B_i; $\Omega(P, B)$ should satisfy the following conditions.

All defined regions by interior constraints are in the defined region by exterior constraints; namely, $r_i \subset r_0$.

The mutual parts of feature constraints are the finite points in collection P; namely, $B_i \cap B_j \subset P$, where $0 \le i \ne m \le j$.

For any $\Omega(P, B)$ with the above conditions, inserting some additional points on the features, how can we get a geometrical equivalent DT graph by using the whole discrete data and additional points corresponding to former graph? Two types of refinement algorithms for conforming DT are considered for solving the problem. One is refining feature lines at first and then executing DT, such as in the literature of [10–12]. Others have the different ideas exactly. Firstly they execute DT and then refine the feature lines. References [10, 13–17] are the representative algorithms.

Inspired by the algorithms of [14, 16, 17] an improved feature refinement algorithm named endpoint triangle's circumcircle model (ETCM) is proposed in this paper.

3. Endpoint Triangle's Circumcircle Model

3.1. Basic Idea of ETCM Definition. Endpoint's triangle containing a feature is such a triangle which uses one of the endpoints as a vertex and intersects one edge of the triangle with the feature simultaneously.

Suppose $s[P_h P_e]$ is a feature that is not contained in DT meshes, Δ_h is the endpoint's triangle of P_h, and Δ_e is the endpoint's triangle of P_e; $\text{Cir}(\Delta_h)$ and $\text{Cir}(\Delta_e)$ are the circumcircles of Δ_h and Δ_e, respectively. The basic idea of ETCM is as follows.

Let $J_h = \text{Cir}(\Delta_h) \cap s[P_h P_e]$, $J_e = \text{Cir}(\Delta_e) \cap s[P_h P_e]$. If $J_h \cap J_e = \varnothing$, choose the longer one between J_h and J_e as the line to be inserted; the corresponding point of intersection is as additional point. Then let the shorter one be the remainder feature line; execute the approach to the shorter one as mentioned above. It will stop execution until $J_{h'} \cap J_{e'} \ne \varnothing$ and take the midpoint of $J_{h'} \cap J_{e'}$ as additional point at that time. In particular if the feature line being treated influences the feature line which has been inserted, the influenced feature line segment should be transacted again.

3.2. Description of ETCM. For designing the ECTM algorithm, we must confirm a data structure at first. There are mainly four structures being considered as shown in Algorithm 1.

```
        Triangle
int index;
intidx_v[3];
bool cross;
Triangle * neighbourtri[3];
Triangle * front;
Triangle * next;
        FeatureSegment
int index
int idx_vb;
int idx_ve;
FeatureSegment * front
FeatureSegment * next;
        TriIndex
Triangle * pTri;
TriIndex * front;
TriIndex * next;
        Vertex
int index;
Int x;
int y;
TriIndex * neighbourtri;
```

Algorithm 1: Data structure of ETCM.

The structure *Triangle* records the information of triangles, including the index number, the index of three vertexes, the tag of intersection, the pointers of three neighboring triangles, and the pointers indicating front or next in the linked list. The structure *TriIndex* is a doubly linked list of triangle. The structure *FeatureSegment* records the information of constraint feature, including the index number, the index of start point and endpoint, and the pointers indicating front or next in the linked list. The structure *Vertex* records the information of vertex, including the index number, the vertex coordinate, and the head pointer of neighboring triangles list. The data structure definitions are shown in Algorithm 1.

The overview of the ECTM algorithm is as shown in Algorithm 2.

The function InsertNewNode(*crosstri*, P_a) is described as shown in Algorithm 3.

For the ETCM algorithm, it can ensure the distribution of additional points is unique. Due to the fact that the additional point is in the influence polygons of feature segment, we can search the additional points in the influence polygons instead of in the whole domain. So it has a high-level efficiency compared to other algorithms.

4. Convergence of the Number of Additional Points

In the process of dealing with the feature line, take the point corresponding to the longer one in the two interceptive segments, which is generated by the two endpoints' circumcircles and feature line as the adding point. It is the point that makes the ratio largest between the interceptive segment and the whole feature line and constructs two triangles which share the interceptive segments and satisfy DT property. By dealing

```
Input parameters: DT meshes and feature lines.
Step 1. Construct a stack FS_remain for storing the features waiting to be treated, and a doubly
        linked list FS_finished for storing the treated features.
Step 2. While FS_remain is not empty, popup a feature segment $s[P_iP_j]$.
Step 3. If $P_j$ belongs to the vertexes of the neighbor triangle of $P_i$, continue.
Else
Step 4. Find the triangles intersected with $s[P_iP_j]$ by the topology relationship of meshes, and
        construct the influence polygon of $P_iP_j$ named crosstri.
        Assign the value to the endpoints of feature segment: $P_h = P_i$, $P_e = P_j$, and $\mathrm{Cir}(\Delta_h)$
          intersects $s[P_hP_e]$ at $Q_1$, $\mathrm{Cir}(\Delta_e)$ intersects $s[P_hP_e]$ at $Q_2$.
Step 5. While $(|P_hQ_1| + |Q_2P_e| < |P_hP_e|)$
    Step 5-1. If $(|P_hQ_1| \geq |Q_2P_e|)$, take $Q_1$ as adding point, i.e. $P_a = Q_1$.
              link $s[P_hQ_1]$ into FS_finished.
              Take $P_a$ as the head of the remainder, $P_e$ as the tail of the remainder.
    Else
    Step 5-2. Take $Q_2$ as the adding point, i.e. $P_a = Q_2$.
              link $s[Q_2P_e]$ into FS_finished.
              Take $P_a$ as the tail of the remainder, $P_h$ as the head of the remainder.
    Step 5-3. InsertNewNode(crosstri, $P_a$)
    Step 5-4. Update $Q_1$, $Q_2$
              $\mathrm{Cir}(\Delta_h)$ intersects $s[P_hP_e]$ at $Q_1$, $\mathrm{Cir}(\Delta_e)$ intersects $s[P_hP_e]$ at $Q_2$.
Step 6. Take the midpoint of $Q_1Q_2$ as additional point $P_a$, link $s[P_hP_a]$ and $s[P_aP_e]$ into
        FS_finished, and the remainder of feature segment is empty.
Step 7. InsertNewNode(crosstri, $P_a$) and return to Step 4.
```

Algorithm 2

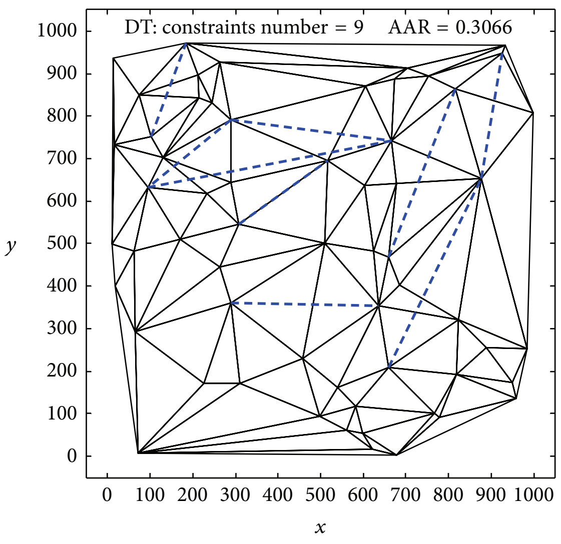
```
Step 1. Find the triangle $\Delta P_a$ or quadrangle $\square P_a$ which includes $P_a$ in the triangle marked by crosstri.
Step 2. Search the triangles whose circumcircle contains $P_a$ iteratively, construct the influence
        polygons of $P_a$ named efftri, and judge whether $P_a$ is contained in a triangle by the sign of area coordinates.
Step 3. If the sharing edge of two triangles in efftri is the feature segment having been treated,
        take this edge out of FS_finished, and push it into FS_remain.
Step 4. Connect $P_a$ with the vertexes of the influence polygons for constructing new triangles.
        Replace the triangle recorded in efftri, then add the remainder of new triangles to the tail of the triangle array.
Step 5. Update crosstri
    Step 5-1. crosstri = crosstri − (crosstri ∩ efftri);
    Step 5-2. If $(s[P_hP_e] \cap s[C_kC_{k+1}] \neq \varnothing$, where $C_k$ and $C_{k+1}$ are two neighbor vertexes of influence polygons of $P_a)$
    Step 5-3. If $(P_a = Q_1)$ $\Delta_h = \Delta P_aC_kC_{k+1}$, add $\Delta_h$ to the head of crosstri.
Else
    Step 5-4. $\Delta_e = \Delta P_aC_kC_{k+1}$, add $\Delta_e$ to the tail of crosstri.
Step 6. Update topology relationship of triangles by the method in [18].
```

Algorithm 3

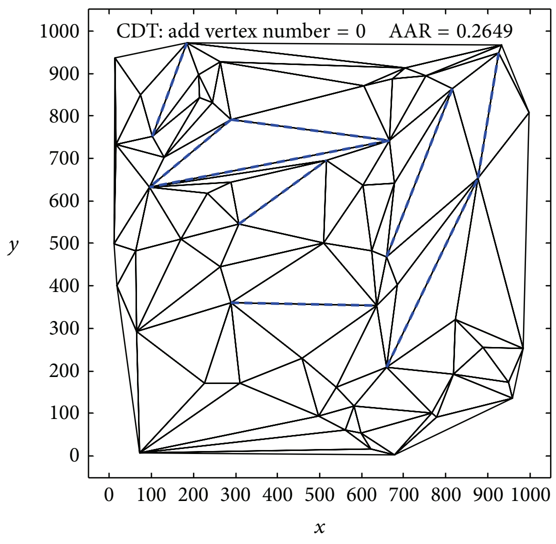
with the remainder feature line as the above operation, until the intersection of two interceptive segments is not empty, there is no doubt that the quantity of additional points for each single feature line would be least compared to any other method. But for the whole planer straight line graph, it is not true in fact. Because of some triangle taking the treated feature segment as its edge, that may become a part of the influence polygon of the feature segment being processed. So the influenced feature segment should be treated again, which would produce some new additional points.

Assume that, in a planer straight line graph with N points and M edges, AB is a feature line, $\mathrm{Circle}(A)$ is the largest circle comprising no other points and features except A, and $\mathrm{Circle}(B)$ has the same definition as $\mathrm{Circle}(A)$. If $\mathrm{Circle}(A) = \mathrm{Circle}(B)$, stop refining AB. If $\mathrm{Circle}(A) \neq \mathrm{Circle}(B)$, $\mathrm{Circle}(A) \cap AB = A'$, $\mathrm{Circle}(B) \cap AB = B'$, the length of $A'B'$ is D, and d is the smallest distance between $A'B'$ and other points or edges, divide $A'B'$ into $[D/2d]$ segments; each segment length is no longer than $2d$. So the circle which takes each segment as its diameter has the character of containing no other points inside. Including AA' and BB', $[D/2d] + 1$ new points would be added on AB, that is, $O(1)$. Because the number of additional points for a feature line does not influence the embedded feature line, the number of all additional points is $O(N)$ at best.

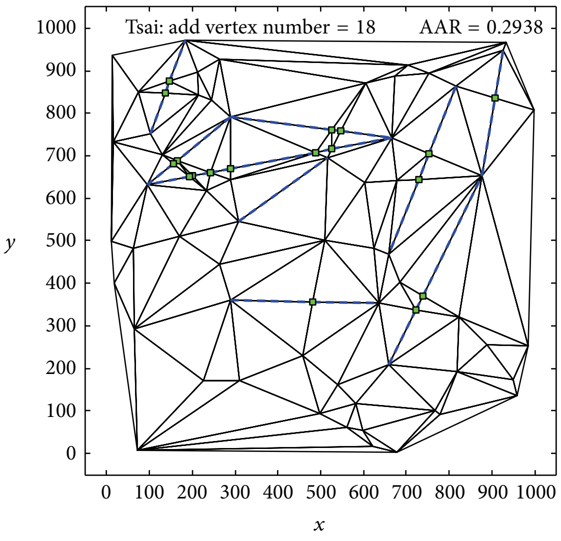
Actually, it has redundancy in refining $A'B'$ by using the method mentioned above. In ECTM algorithm, it only deals with the influenced polygons. So the average radius of



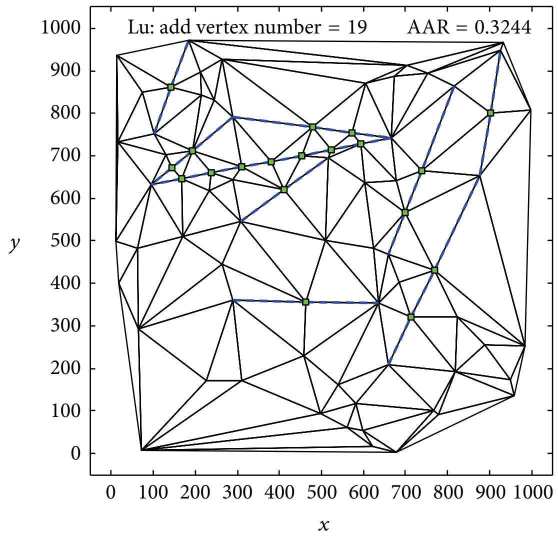
(a) DT



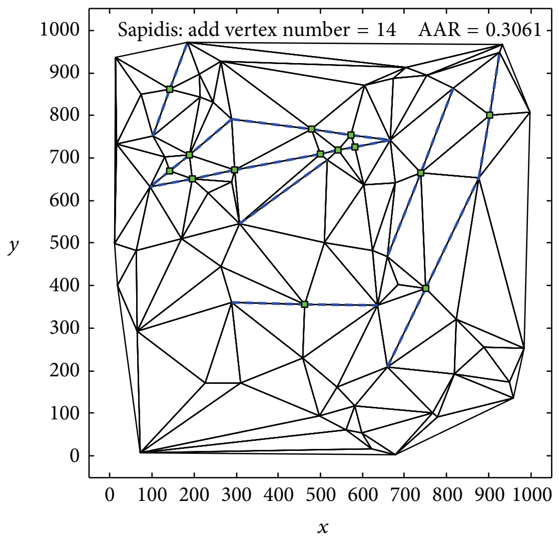
(b) CDT



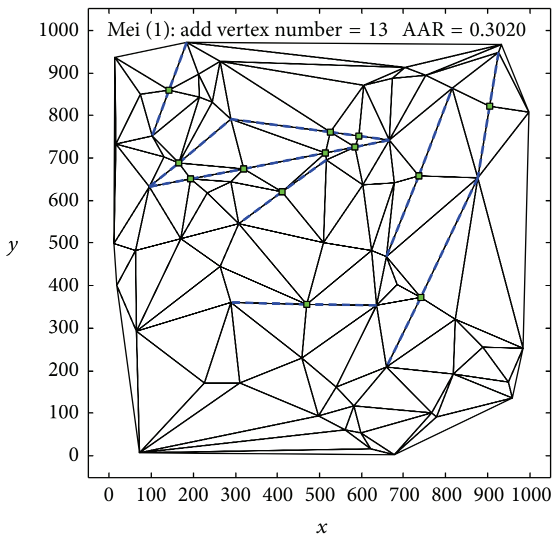
(c) Algorithm Tsai



(d) Algorithm Lu

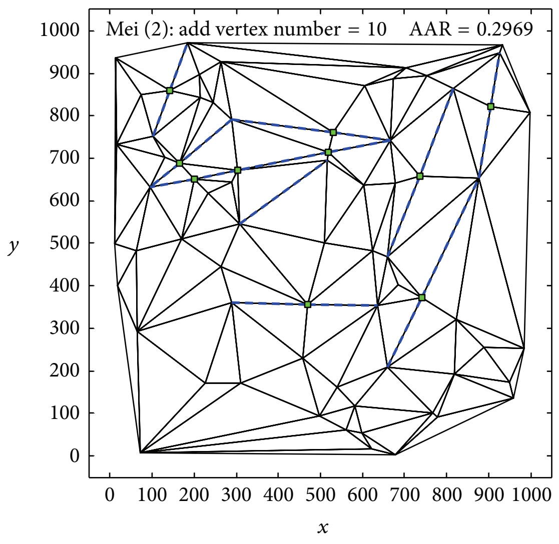


(e) Algorithm Sapidis

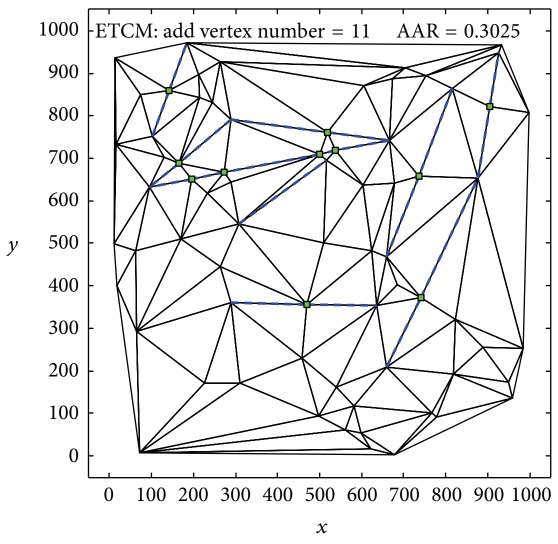


(f) Algorithm Mei (1)

Figure 1: Continued.



(g) Algorithm Mei (2)



(h) ETCM

FIGURE 1: Triangulation results of node refinement algorithms with 60 nodes and 9 feature segments.

the gained Circle(P_i) is larger, and the number of additional points is smaller. Thus the number of additional points by using ETCM algorithm is convergent.

5. Time Complexity of ETCM

Suppose that T_j^c is the time cost for calculating the position of the jth additional point and T_j^i is the time cost for inserting the jth additional point. It was pointed out in literature [19] that the average neighboring triangle number with a point is about six in a DT mesh. So for finding an endpoint's triangle, averaged six intersection judgments would be needed, and then the second triangle intersected with the feature line can be confirmed in the meantime. If a feature line crosses with n triangles, averaged $6 + (n - 2) \times 2$ intersection judgments are required to the influence range of the feature line, and the time cost is a constant for calculating the intersection point between a circumcircle and a feature line; that is, $T_j^c = O(1)$. So for finding the triangle or quadrangle which contains additional point, $n - 1$ times of judgments whether a point is included in a triangle are needed, and the time cost of each judgment is also a constant. In literature [9], it was shown that that is a linear process, so $T_j^i = O(1)$. The time complexity can be calculated as follows:

$$\begin{aligned} T^c &= \sum_{i=1}^{N} T_j^c \leq N \times \max\left\{T_j^c\right\} \Longrightarrow T^c = O(N) \\ T^i &= \sum_{i=1}^{N} T_j^i \leq N \times \max\left\{T_j^i\right\} \Longrightarrow T^i = O(N). \end{aligned} \tag{1}$$

So

$$T_{\mathrm{ECTM}} = T^i + T^c = O(N), \tag{2}$$

where T^c is the time cost for calculating the positions of all additional points and T^i is the time cost for inserting all additional points.

6. Experiments Results and Analysis

6.1. Assessment Criteria. There are no criteria for evaluating the performance of refinement embedding algorithms in the existing references. So we give five assessment criteria as follows:

(1) stability of refinement;

(2) number of additional nodes, which means that, by using as small number of additional points as possible, we can change the original datum set as little as possible;

(3) time cost;

(4) the quality of the mesh after intruding additional nodes: when AAR is 0.5 approximately, it indicates a result of fine quality;

(5) the average aspect ratio (AAR) improved by single additional point. The model is given by the following equation:

$$\mathrm{AAR}_p = \frac{\left(\mathrm{AAR}_{\mathrm{refine}} - \mathrm{AAR}_{\mathrm{CDT}}\right)}{n}, \tag{3}$$

where AAR is the average of all aspect ratios of meshes, $\mathrm{AAR}_{\mathrm{refine}}$ is the AAR after refinement, $\mathrm{AAR}_{\mathrm{CDT}}$ is the AAR after constrained DT, and n is the number of additional points.

6.2. Performance Test. The performance tests were performed for all refinement algorithms previously described

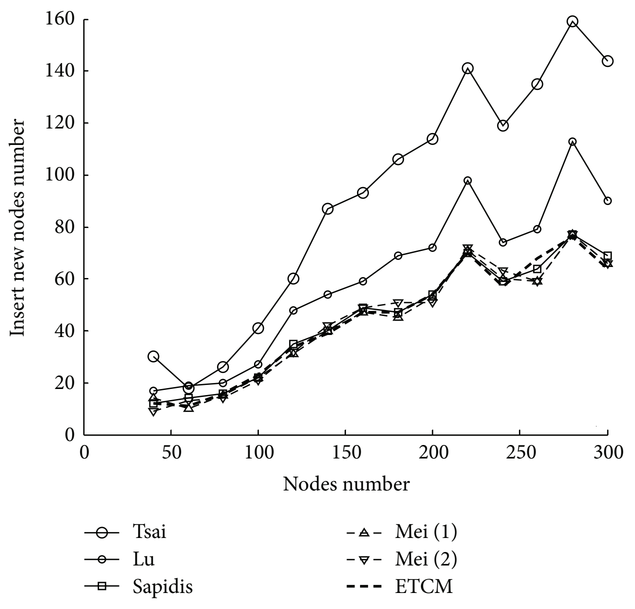


FIGURE 2: Number of new nodes.

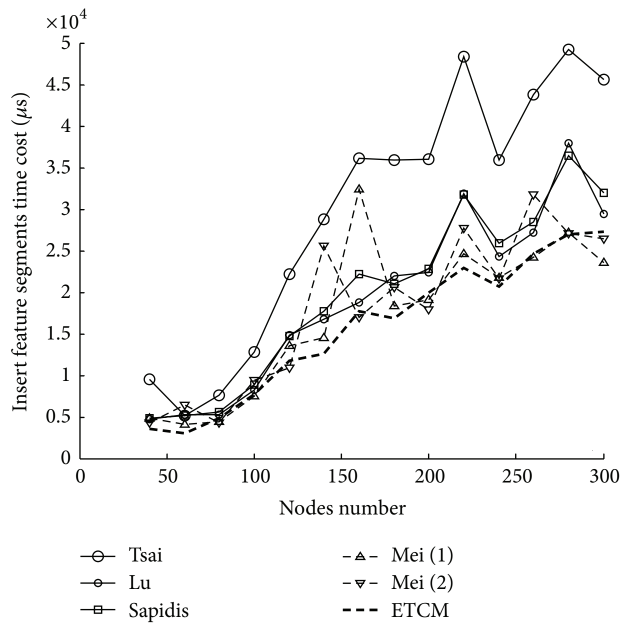


FIGURE 3: Time cost of inserting feature segment.

including Tsai [15], Lu et al. [13], Sapidis and Perucchio [14], Wei et al. [17], and ETCM.

Take 15 planer straight line graphs as testing datum, which comprises a group uniformity distribution random points of each. The coordinates of the points are all in [0, 1000], and the number of feature lines is 15% of the number of the points. The testing environment is on a computer with P4 2.8 G, 256 × 2 M RAM, Windows XP, and Visual C++ 6.0, and the time cost testing tool is CodeTest4.0.

Figure 1 shows a group of resulting mesh of a planer straight line graph, where the thick dashed lines are feature

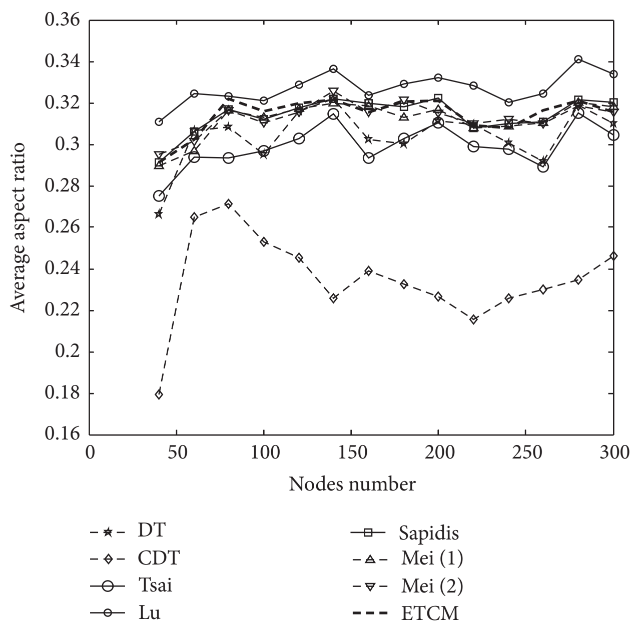


FIGURE 4: AAR of node refinement algorithms.

lines and square points are additional points. Figures 2 to 5 show the performance evaluating results. For testing stability of refinement, we adopt the method that exchanges the two endpoints of each feature line and then judge whether the DT results are conformable.

In the refinement stability test result as Figure 1, all algorithms have unique result except algorithm Mei. It indicates that, except Mei, all the other algorithms are stable. Figure 2 shows that method in which Tsai requires the largest number of additional nodes, methods Lu and Sapidis take the second place, and methods Mei and ETCM need nearly the same least number. From Figure 3, we can conclude that method Tsai's time cost is the biggest and ETCM cost the least time. In particular, ECTM's time cost curve is with the least undulation, which shows a nearly linear relationship with the number of points. It indicates that the time cost of ETCM is mainly affected by points' number and hardly affected by the distribution of the points. The robustness of ETCM is best. The AAR results are shown in Figure 4; all refinement algorithms can make the AAR of the meshes resume to or be better than the AAR of meshes before inserting feature lines. Method Lu has the most obvious improvement, Tsai's effect is the worst, and the other three methods are almost at the same level. Figure 5 shows the AAR improved by single additional point. Method Tsai has the worst capability, Lu is a little better, and the others are pretty much the same thing.

7. Conclusions

Based on the performance analysis of the DT algorithm, a new refinement algorithm named endpoint triangle's circumcircle model (ETCM) is proposed in this paper. The performance analysis is given and confirmed by simulations. Simulation results show that ETCM achieves as well as

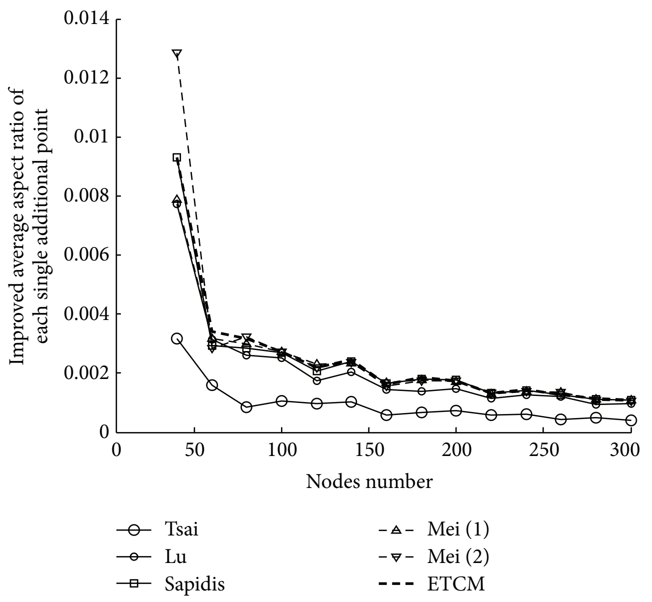


Figure 5: Improved AAR of each new node.

the best in refinement stability, additional point number, and mesh quality. In particular, the time cost of ETCM is minimal, and it is influenced by the data distribution least. Hence, it can be concluded that the proposed refinement technique ECTM can be used to solve some practical problems with faster requirement by maintaining good quality of solution.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 51379049), the Fundamental Research Funds for the Central Universities of China (no. HEUCFX41302), and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, Heilongjiang Province (no. LC2013C21).

References

[1] K. Derr and M. Manic, “Wireless sensor network configuration-part I: mesh simplification for centralized algorithms,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1717–1727, 2013.

[2] J. R. Shewchuk and S. Hang, “Higher-quality tetrahedral mesh generation for domains with small angles by constrained delaunay refinement,” in *Proceedings of the 30th Annual Symposium on Computational Geometry*, pp. 290–299, 2014.

[3] M. S. Ebeida, S. A. Mitchell, A. A. Davidson, A. Patney, P. M. Knupp, and J. D. Owens, “Efficient and good Delaunay meshes from random points,” *Computer-Aided Design*, vol. 43, no. 11, pp. 1506–1515, 2011.

[4] M. Qi, T.-T. Cao, and T.-S. Tan, “Computing 2D constrained delaunay triangulation using the GPU,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 5, pp. 736–748, 2013.

[5] J. R. Shewchuk, “Reprint of: delaunay refinement algorithms for triangular mesh generation,” *Computational Geometry: Theory and Applications*, vol. 47, no. 7, pp. 741–778, 2014.

[6] M. V. Anglada, “An improved incremental algorithm for constructing restricted delaunay triangulations,” *Computers and Graphics*, vol. 21, no. 2, pp. 215–223, 1997.

[7] L. X. Li and J. R. Tan, “Multiple diagonal exchanging algorithm for inserting constrained boundary in constrained Delaunay triangulation,” *Chinese Journal of Computers*, vol. 22, no. 10, pp. 1114–1118, 1999.

[8] A. J. Hansen and P. L. Levin, “On confirming Delaunay mesh generation,” *Advances in Engineering Software*, vol. 14, no. 2, pp. 129–135, 1992.

[9] J. Ruppert, “A Delaunay refinement algorithm for quality 2-dimensional mesh generation,” *Journal of Algorithms*, vol. 18, no. 3, pp. 548–585, 1995.

[10] J.-D. Boissonnat, “Shape reconstruction from planar cross sections,” *Computer Vision, Graphics and Image Processing*, vol. 44, no. 1, pp. 1–29, 1988.

[11] O. D. Faugeras, E. Le Bras-Mehlman, and J.-D. Boissonnat, “Representing stereo data with the Delaunay triangulation,” *Artificial Intelligence*, vol. 44, no. 1-2, pp. 41–87, 1990.

[12] H. Edelsbrunner and T. S. Tan, “An upper bound for conforming delaunay triangulations,” *Discrete & Computational Geometry*, vol. 10, no. 2, pp. 197–213, 1993.

[13] Z. Y. Lu, C. K. Wu, and X. N. Zhou, “Overall delaunay triangulation of 2D scattered data with characteristic constrains,” *Chinese Journal of Computers*, vol. 20, no. 2, pp. 118–124, 1997.

[14] N. Sapidis and R. Perucchio, “Delaunay triangulation of arbitrarily shaped planar domains,” *Computer Aided Geometric Design*, vol. 8, no. 6, pp. 421–437, 1991.

[15] V. J. D. Tsai, “Delaunay triangulations in TIN creation: an overview and a linear-time algorithm,” *International Journal of Geographical Information Systems*, vol. 7, no. 6, pp. 501–524, 1993.

[16] F. L. Yi and D. Z. Han, “Delaunay triangulation dividing optimal algorithm with constraint line,” *Computer Engineering*, vol. 27, no. 6, pp. 32–34, 2001.

[17] C.-L. Wei, G.-Y. Xiao, and Y.-H. Zhou, “New algorithm for conforming Delaunay triangulation,” *Acta Electronica Sinica*, vol. 29, no. 7, pp. 895–898, 2001.

[18] X. J. Liu and X. S. Fu, “A study of algorithm for constructing triangulation irregular net (TIN),” *China Journal of Highway and Transport*, vol. 13, no. 2, pp. 31–36, 2000.

[19] O. R. Joseph, *Computational Geometry in C*, Cambridge University Press, Cambridge, UK, 2nd edition, 1998.