A New Algorithm for Guarding Triangulated Irregular Networks*

Alireza Zarei
zarei@mehr.sharif.edu

Mohammad Ghodsi
ghodsi@sharif.edu

Computer Engineering Department, Sharif University of Technology
P.O. Box 11365-9517, Tehran, IRAN
IPM School of Computer Science, P.O. Box 19395-5746, Tehran, IRAN

Abstract: In this paper, we present a new algorithm for vertex guarding of a triangulated surface, which takes into account the heights of the vertices and considers the global visibility of the guards. In this algorithm, the initial surface is reduced to a collection of simple polygons and *trivial* triangulated surfaces. This is done by assigning guards to some vertices and removing the faces covered by these guards. The remaining simple regions are then guarded properly. The running time of our algorithm is linear in terms of *n*, the number of vertices. The upper bound of the assigned guards is $\lfloor 2n/3 \rfloor$, and its expected number is $\lfloor n/2 \rfloor$, which is the same as the worst case lower bound for the problem. The running time of the best known algorithm for this problem is $O(n^{3/2})$ which selects at most $\lfloor n/2 \rfloor$ guards. We expect a better performance of our algorithm on the average in practice. This has been verified by experimentation.

Keywords: NP-completeness, art gallery problem, guard set, triangulated irregular networks, approximation algorithms

1 Introduction

In 1973, Victor Klee asked this question ``How many guards are required to guard an art gallery?" This was named as the *art gallery problem* and many of its different versions have been considered by researchers [1]. We focus on solving this problem in 3-dimensional surfaces. To make the problem easier, we assume that each vertical line intersects the surface in at most one point. Such surfaces are known as 2.5-dimensional surfaces. There are several computational models for such regions. One, that is used often, is to approximate the region by a collection of joint triangular faces. This model, known as TIN (Triangulated Irregular Networks), is used in this paper.

Vertex guarding on TINs is defined as selecting the minimum number of vertices (or guards), such that each point of the surface is visible from at least one of the guards. Two points on a TIN are visible to each other if their connecting segment does not intersect any other point of the surface and their connecting segment lies above the TIN.

This problem is known to be NP-hard [2]. Therefore, attempts have been focused on finding reasonable approximations [3, 4]. Nevertheless, a wide range of these problems are not approximable within a constant factor [5]. So, the main contribution is to come up with an efficient approximation that guarantees the worst case bound.

Moreover, visibility processing in 2.5-dimensional surfaces is still complicated. Therefore, a simpler version has been considered in most solutions in which the heights of the points are all ignored. In such simplification, a vertex can only cover its adjacent faces. This reduces the problem to a 2-dimensional version and the TIN can be represented by a planner graph with triangular faces.

* This work has been supported by a grant from IPM school of CS (No. CS1384-2-01).

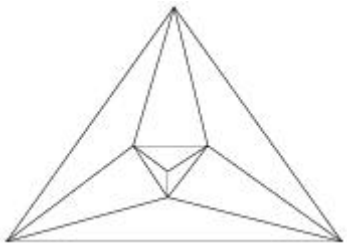

Fig. 1: A TIN which requires at least $\lfloor n/2 \rfloor$ vertex guards.

According to the four-color theorem, $\lfloor n/2 \rfloor$ guards are always sufficient to guard a TIN with n vertices and as shown in Fig. 1, this number of guards is sometimes necessary [6]. Based on this idea, an $O(n^2)$ time algorithm have been presented which uses at most $\lfloor n/2 \rfloor$ guards [7].

Another approach is based on finding the maximum matching in the dual graph of a planar graph which leads to an $O(n^{3/2})$ time algorithm to assign at most $\lfloor n/2 \rfloor$ guards [7]. There are faster and easier algorithms at expense of more guards; an example is a linear time algorithm that selects at most $\lfloor 3n/5 \rfloor$ guards [6]. In this algorithm, the corresponding planar graph is colored by five colors and then the vertices colored by one of the least used three colors are selected as guards.

In this paper, a new approximation algorithm is presented that runs in linear time and selects at most $\lfloor 2n/3 \rfloor$ guards in the worst case. The expected number of the guards is $\lfloor n/2 \rfloor$, which is the same as the lower bound of this problem.

Despite its predecessors, our algorithm takes the global visibility of vertices into account. In our proposed algorithm the heights of the vertices are considered to determine whether a vertex can see its non-adjacent faces. We do this during the incremental process of guard assignment without increasing the order of the algorithm. This height consideration reduces the upper bound of the average number of the assigned guards to $\lfloor n/2 \rfloor$. Moreover, if we pay more costs to prepare the faces visible from each vertex, which requires $O(n^3)$ of preprocessing time, the number of the assigned guards is significantly reduced. This has been verified by experimentation. On the other hand, the optimal number of the guards is usually less than $\lfloor n/2 \rfloor$. Therefore, our algorithm can produce better results on real surfaces.

In the rest of this paper and in Section 2, guarding problem for a collection of simple polygons is considered. In Section 3, we present a method by which a TIN is reduced to a set of such polygons or trivial TINs. In Section 4, the new algorithm is proposed and its average case analysis is described in Section 5. Finally, the materials are summarized and concluded in Section 6.

2 Guarding Simple Polygons

There is a linear time algorithm that selects at most $\lfloor n/3 \rfloor$ vertex guards covering any n vertex triangulated simple polygon (without holes) [8].

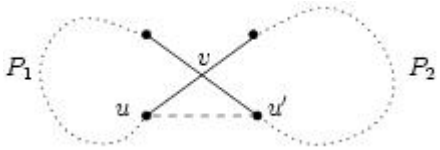

Fig. 2: The connected polygons P_1 and P_2 can be extended to a single one by adding the edge uu' and removing the edges uv and $u'v$.

Trivially, this algorithm is applicable on a collection of triangulated disjoint polygons P_1, P_2, …, P_k. We show that this result can also be applied to a set of polygons that may have common vertices but in such a way that their exterior region remains connected.

According to Fig. 2, it is possible to combine two joint polygons and produce a single simple polygon. This is done by adding the edge uu' that connects two adjacent vertices of the common vertex v, one from each polygon, and then removing the edges uv and $u'v$ from the resulting polygon. Trivially, this conversion does not change the number of the vertices. Doing this conversion iteratively, any collection of simple polygons $\boldsymbol{S}=\{P_1, P_2, \ldots, P_k\}$ whose exterior region is connected can be converted to a collection $\boldsymbol{S'}$ of disjoint simple polygons. The number of vertices of polygons in $\boldsymbol{S'}$ is equal to the number of the vertices of polygons in $\boldsymbol{S}$ and any set of guards that covers the interior of $\boldsymbol{S'}$ is also sufficient to cover the interior of $\boldsymbol{S}$. Consequently,

Theorem 1. A set P_1, P_2, …, P_k of simple polygons, whose exterior region is connected (has no hole), can be guarded by at most $\lfloor n/3 \rfloor$ vertex guards in $O(n)$ time in which n is the number of the vertices of all P_i's (common vertices of polygons are enumerated only once).

3 Reducing a TIN to a Set of Simple Polygons and Trivial TINs

The main idea of our algorithm is to make the initial TIN simpler by incrementally selecting a set of vertices (as guards) and removing those faces that are visible to these guards. In this section, we present this reduction process that reduces the TIN to a set of simple polygons and *trivial* TINs. We will use the result of the previous section to guard the remaining faces. Therefore, this reduction process must be done in such a way that the exterior region of the remaining faces remains connected.

To achieve this goal and to assign the guards efficiently, two properties must exist:

1. Removing a vertex and its dominated faces must not create any hole inside the remaining TIN.
2. Any removed vertex (which is a guard) must have at least four adjacent triangles.

The first property is satisfied by using the following strategy during the reduction process; when a guard is assigned to a vertex, only that connected set of the visible faces around the guards are removed from the TIN.

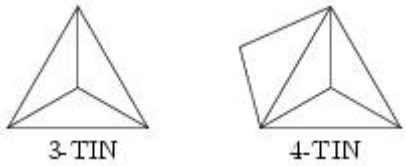

Fig. 3: Trivial TINs do not have a vertex with more than three adjacent faces.

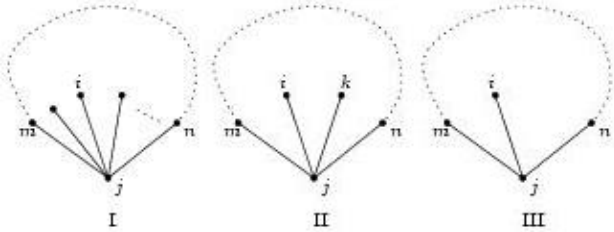

Fig. 4: Different possibilities of an internal vertex *i* and its boundary vertex *j* depending on the degree of *j*: I) *j* has at least four adjacent faces, II) *j* has three adjacent faces, and III) *j* has two adjacent faces.

In addition, a guard is only assigned to a vertex that its covered region is adjacent to the exterior one in at least one edge. Hence, this type of vertex and face deletion does not create any hole inside the TIN. None of the TIN's shown in Fig. 3 has a vertex that supports the latter property. We call these TINs trivial and are referred to as 3-TIN and 4-TIN in the rest of paper. We show that any non-trivial TIN is either a simple polygon or has a vertex with at least four adjacent triangles and its removal satisfies both of the properties.

In any TIN which is not a simple polygon, there is at least one none boundary vertex *i*. If *j* is a boundary vertex adjacent to *i*, depending on the degree of *j*, there are three possibilities shown in Fig. 4:

I) The boundary vertex *j* has at least four adjacent triangles that are removed from the TIN by removing *j*, and this does not create any hole inside the TIN. Therefore, both of the properties are satisfied here.

II) *j* has three adjacent faces. If one of the *i* or *k* vertices has more than three adjacent triangles, removing it will satisfy the second property and because of the removal of the *jn* or *jm* edges it does not create any hole inside the TIN. Hence, it satisfies the first property, too.

If none of the *i* and *k* has more than three adjacent triangles, the TIN must be the same as a 4-TIN. The reason is that, as assumed, *i* is an internal vertex of the TIN and has at least three adjacent triangles. The triangles Δijk and Δmji are already adjacent to *i*. So, the only remaining triangle that can be adjacent to *i* is Δmik (see Fig. 5). This will also fix the state of *k* with three adjacent triangles (Δjkn, Δijk, Δmik). Therefore, if the TIN has more vertices than a 4-TIN, it had to be extended from *n* or *m* as shown in Fig. 5. However, the middle TIN is a 4-TIN.

III) *j* has only two adjacent faces. If any one of *i*, *n* or *m* vertices has more than three adjacent triangles, selecting it as guard will satisfy both of the properties. But if none of them has, the TIN must be the same as a 3-TIN or a 4-TIN as shown in Fig. 6.

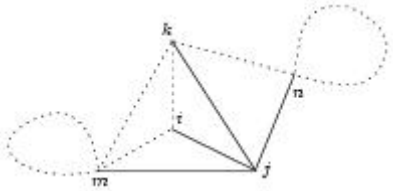

Fig. 5: For the second possibility of Fig. 4, when none of *i* and *k* has more than three adjacent triangles, the TIN must be a 4-TIN.

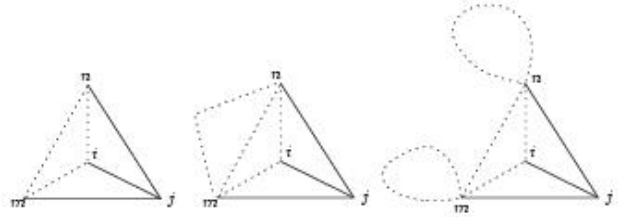

Fig. 6: For the third possibility of Fig. 4, when none of *i*, *n* and *m* has more than three adjacent triangles, the TIN must be a 3-TIN or a 4-TIN.

Consequently,

Theorem 2. Each TIN can be reduced to a collection of simple polygons and trivial TIN's by removing several of its vertices so that during the reduction process, the exterior region of the remaining faces remains connected and any one of the removed vertices (assigned by a guard) has at least four adjacent faces that will also be removed from the TIN.

4 The Proposed Algorithm

Our proposed algorithm has three phases:

1) In the first phase, as described in Section 3, the initial TIN is converted to a collection (chain) of simple polygons and trivial TIN's. This is done by assigning guards to some vertices of the TIN and removing the faces covered by these guards. This phase is continued until there is no vertex with more than three adjacent faces.
2) In the second phase, the internal vertex of each 3-TIN receives one guard. Also, one of the boundary vertices of each 4-TIN, which has three adjacent faces, receives one guard.

   After this phase, only a collection of simple polygons remains each of which Theorem 1 is applied.
3) Finally, the remaining faces are covered with the method described in section two.

In order to improve the efficiency of the algorithm, when a new guard is assigned and its covered faces are removed, the newly appeared faces on the boundary of the remaining TIN are checked against this guard. Any one of these faces that is visible from the new guard is also removed from the TIN. This process is continued until there was no such newly appeared face on the boundary of the TIN. Every face can be appeared on the boundary at most three times from any one of its edges. Therefore, this improvement takes linear time in terms of the number of the faces of the TIN. However, as

shown in the next section, this strategy reduces the expected number of the guards to $\lfloor n/2 \rfloor$.

In the following subsections, the correctness of the algorithm is proved and its efficiency and running time complexities are analyzed.

4.1 The algorithm correctness

It is simple to prove that this algorithm covers all faces of the TIN: In each step of the first phase, only the faces that are visible to a vertex are removed and because a guard is assigned to that vertex the removed faces are guarded correctly. After reduction phase, a new guard is assigned to each 3-TIN which covers it completely. Also, a new guard is assigned to each 4-TIN which covers all faces of it, except one face which will be added to the collection of the remaining simple polygons.

Finally, by using the method described in section two, the remaining simple polygons will be guarded. Therefore,

Theorem 3. The set of the selected guards of the algorithm covers all faces of the TIN.

4.2 Upper bound of the number of the guards

We first introduce several basic lemmas:

Lemma 1. After the reduction phase, there is no common vertex among 3-TINs.

Proof. If there is such a vertex, it must have more than three adjacent faces. But, vertices with more than three adjacent faces are removed from the TIN by assigning a guard to them during the reduction phase.

■

Lemma 2. After the reduction phase, if a 3-TIN and a 4-TIN have a common vertex, all of the adjacent faces of that common vertex are covered during the second phase of the algorithm. Therefore, it will be removed from the TIN automatically.

Proof. Such a vertex has two adjacent faces from the 3-TIN and one from the 4-TIN. According to the method of assigning guard to the trivial TINs, after the second phase of the algorithm, all these faces are covered and removed from the TIN. Therefore, after the second phase, that vertex has no adjacent faces and will be removed from the TIN automatically.

■

Assume that K is the set of the 3-TINs produced after the first phase of the algorithm and $k=|K|$. Also denote by I the set of the remaining faces of the TIN, after the second phase, which at least one of the boundary vertices of any one of them used to be a boundary vertex of a 3-TIN and $i=|I|$. If $i \le k$ then the following lemma exists:

Lemma 3. The number of the boundary vertices of the 3-TINs which have no adjacent faces after the second phase of the algorithm and will be removed from the initial TIN automatically is greater than or equal to $3k-3i$.

Proof. Each 3-TIN has three boundary vertices and no one of them is common between two such TINs. Hence, there are $3k$ such vertices. On the other hand, any one of the faces in I can be adjacent to at most three of these vertices. Therefore, at least $3k-3i$ vertices of them are not adjacent to any one of the remaining faces and can be removed without requiring any guard on them.

■

These lemmas help us to prove the following theorem about the upper bound of the number of the guards selected by our algorithm:

Theorem 4. The upper bound of the number of the selected guards of the algorithm, for a TIN of n vertices, is $\lfloor 2n/3 \rfloor$.

Proof. Denote by m the number of the assigned guards during the reduction phase and by j the number of the 4-TINs after the reduction phase.

Each TIN of n vertices and t boundary ones has $2n-t-2$ faces [9]. In the first phase of our algorithm, when a guard is assigned to a vertex, at least four faces of the TIN are removed from it. Each 3-TIN has three faces and each 4-TIN has four faces and these faces are not common and are distinct from the faces in I. So the following inequality holds:

$$\begin{aligned} & 4m+4j+3k+i \le 2n-t-2 \\ \Rightarrow\ & 4m+4j+3k+i \le 2n \qquad (1) \end{aligned}$$

We illustrate two possibilities:

1) $i \le k$: In this case, according to lemma 3, at least $3k-3i$ vertices will be removed from the TIN without assigning any guard to them. So, the number of the guards is obtained from the following relation($N(G)$ is the number of the guards):

$$N(G) \le m+j+k+\left\lfloor \frac{n-(m+j+k)-(3k-3i)}{3} \right\rfloor \quad (2)$$

Relation (1) leads to $m+j \le \frac{2n-3k+i}{4}$ and by substituting in relation (2) the following is resulted:

$$\begin{aligned} N(G) &\le \frac{2n-3k-i}{4}+k+\left\lfloor \frac{n-(\frac{2n-3k-i}{4}+k)-(3k-3i)}{3} \right\rfloor \\ \Rightarrow &\le \frac{6n-9n-3i+12k+4n-2n+3k+i-16k+12i}{12} \\ \Rightarrow &\le \frac{8n-10k+10i}{12} \\ \Rightarrow &\le \frac{2n}{3}+\frac{5}{6}(i-k) \end{aligned}$$

By the assumption that $i \le k$, for this case the upper bound of the number of the guards is $\lfloor 2n/3 \rfloor$.

2) $i > k$: We know that:

$$N(G) \le m+j+k+\left\lfloor \frac{n-(m+j+k)}{3} \right\rfloor \quad (3)$$

Combining relation (1) and the ($i > k$) assumption, leads to $m+j+k \le \frac{2n}{4}$ and by substituting in relation (3) the following is resulted:

$$N(G) \leq \frac{2n}{4} + \left\lfloor \frac{n-(2n/4)}{3} \right\rfloor$$
$$\Rightarrow \leq \frac{6n+4n-2n}{12}$$
$$\Rightarrow \leq \frac{2n}{3}$$

So, the theorem holds in both cases.

■

4.3 The running time analysis

We assume that a DCEL(Doubly-Connected Edge List) structure is used to hold the TIN. In this data structure, navigating the boundary edges of a face and the adjacent faces of a vertex, edge or face can be done in linear time. Other specifications and properties of this data structure exist in [9].

During the reduction phase of the algorithm, each vertex is processed at most once and the running time of removing that vertex and reconstructing the DCEL is proportional to the number of the adjacent faces of it. Therefore, for all vertices of the TIN, the total cost of this process is linear in terms of the number of the vertices of the TIN. Also each face can appear on the boundary of the TIN at most three times from each edges of it. So, the whole running time required to check if a newly appeared face on the boundary of the TIN is visible from the last assigned guard, is linear in terms of the number of the faces of the TIN. On the other hand, depending on the Euler formula, the number of the faces and edges of a TIN is a linear function of the number of the vertices of it [9].

Consequently, the running time of the first phase of the algorithm is $O(n)$ in which n is the number of the vertices of the TIN.

Using the DCEL structure, identifying and guarding the trivial TINs, in the second phase of the algorithm, can be done in linear time. Also, the last phase of the algorithm as said in Theorem 1, is done in linear time.

Theorem 5. The running time of the algorithm is linear in terms of the number of the vertices of the TIN.

5 Average Case Analysis

The average case lower bound of the number of the guards is derived from the following two claims:

1) During the first phase of the algorithm, the expected number of the covered faces by any guard is 6.
2) After the first phase of the algorithm, the expected number of the guards for covering the remaining faces is at most $\lfloor k/3 \rfloor$ in which k is the number of the vertices of these faces.

A face A is called a second-order neighbor of a vertex v if A is not adjacent to v but has some common edges with at least one of the adjacent faces of v. In the following average case analysis, the two probabilities; A is visible to v or not; are assumed to be equal. In fact, these probabilities depend on the heights of v, its adjacent vertices, and vertices of A.

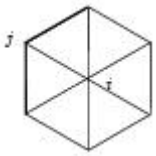

Fig. 7: The expected number of the adjacent faces of a vertex i in a plane graph is 6.

These heights determine the angle between two adjacent faces which may be either less than or greater than 180^o which A will then be visible or invisible from v, respectively. This will help us to prove the claims.

According to the Euler formula, the average degree of a vertex in a plane graph is 6 (see Fig. 7). During the first phase of the algorithm, when a guard is assigned to a vertex i, it will cover 4 or 6 faces of the TIN. This depends on whether another guard has been assigned to any one of its neighbor vertices, like j, before assigning a guard on i.

Also, a vertex can cover some of its non-adjacent faces. There are many different situations for a vertex i with respect to its adjacent and non-adjacent faces that must be considered. However, to prove our claim it is sufficient to show that the expected number of the second-order neighbor faces that will be covered by the newly selected guard is at least one. This is proved by comparing the different situations of vertex i and its neighbors: Only in one state i has no second-order neighbor face and in contrast, there is a state in which i has four such faces as shown in part I of Fig. 8. Also, for any one of the states in which i has only one second-order neighbor face, there is a distinct state that i has three such faces as shown in part II and III of Fig. 8.

Therefore, each vertex i has at least two second-order neighbor faces on the average. According to our assumption, by which a vertex can see its second-order neighbor with probability of $\frac{1}{2}$, the expected number of such faces that will be covered by i is one.

Depending on the existence or absence of the vertex j, on which a guard has been assigned before assigning on i, the expected number of the covered faces is at least 5 or 7, respectively. Therefore, during the first phase of the algorithm, the expected number of the covered faces by any selected guard is at least 6 and this is our first claim.

After the first phase of the algorithm, the remaining unguarded faces of the initial TIN compose a collection of simple polygons and trivial TINs. In section two, we showed the second claim for simple polygons. So, its correctness must be shown only for trivial TINs.

When a guard is assigned to the internal vertex of a 3-TIN, according to part I of Fig. 9, all of its faces will be covered. In addition, anyone of its boundary vertices which is not common with another remaining region will be removed from the TIN without requiring another guard on it. Moreover, assume that such a region, like face D adjacent to vertex d as shown in part I of Fig. 9 exists. If d is visible to the internal vertex a, which currently has a guard, the corresponding boundary vertex (d) does not require any

guard on it and will be removed from the TIN automatically. Therefore, the expected number of the vertices that will be removed from a 3-TIN, when a guard is assigned to its internal vertex, equals $3(\frac{1}{2}+\frac{1}{4})+1$ which is greater than 3.

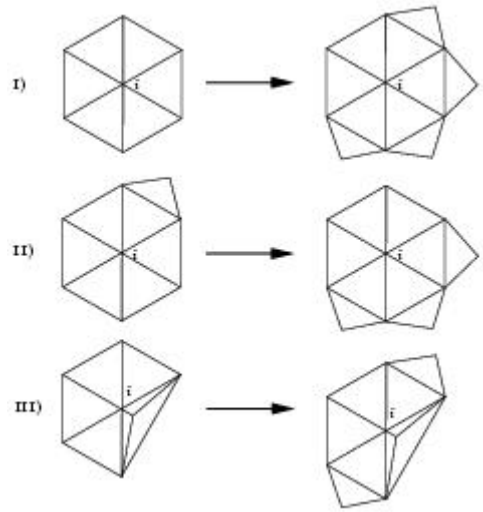

Fig. 8: Correspondence between the different situations of a vertex *i* with respect to the number of its second-order neighbors: I) The corresponding states in which *i* has none and 4 second-order neighbors, II, III) The corresponding states in which *i* has one and 3 second-order neighbors.

On the other hand, when a guard is assigned to a boundary vertex of a 4-TIN, according to part II of Fig. 9, if this guard (*a*) sees the only remaining triangle of the TIN (*A*), *b* and *c* vertices can be removed without requiring any more guards. Also, if the faces *E* and *F* do not exist or they are visible from *a*, the other two vertices, *d* and *f*, will also be removable. There can be two faces like the face *F* adjacent to vertex *f*. So, the expected number of the vertices that will be removed from a 4-TIN, when a guard is assigned to a boundary vertex of it, equals $(1/2+1/4)+((1/2+1/4)/2)+2(1/2)+1$ which is greater than 3.

Consequently, after the first phase of the algorithm, the expected number of the guards required to cover the remaining regions of the initial TIN is $\lfloor k/3 \rfloor$ in which *k* is the number of the vertices of these regions. So, our second claim is also true for all kind of the remaining regions.

Now, we combine the two results obtained so far. Denote by *m* the number of the guards selected in the first phase of the algorithm and by *t* the number of the remaining faces after this phase for an initial TIN of *n* vertices. We have:

$$6m+t \le 2n \quad \Rightarrow \quad m \le \frac{2n-t}{6} \quad (4)$$

$$0 \le t \le n-m \quad (5)$$

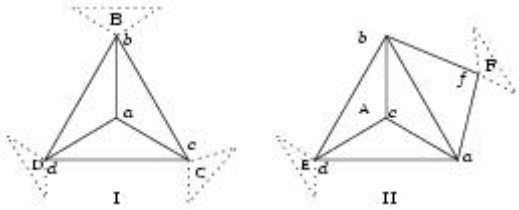

Fig. 9: I) When a guard is assigned to *a*, if the face *D* does not exist or it is visible from *a*, the vertex *d* will be removed from the TIN automatically. This is also true for (*b*, *B*) and (*c*, *C*) pairs. II) When a guard is assigned to *a*, if the face *A* is visible from *a*, the vertices *b* and *c* will be removed from the TIN automatically. Also, depending on the visibility of *E* and *F* from *a*, the vertices *b* and *c* can be removed.

By combining these equations, the upper bound of *m* for the minimum and maximum values of *t* is:

$$t=0: \qquad m \le \frac{2n}{6} \qquad (6)$$

$$t=n-m: \quad m \le \frac{2n-(n-m)}{6} \Rightarrow m \le \frac{n}{5} \quad (7)$$

In the first case, $\lfloor n/3 \rfloor$ guards cover the TIN and in the second case the upper bound of the number of the guards is derived from the following equation:

$$N(G) \le m + \frac{n-m}{3}$$

By substituting *m* in the above equation with its upper bound from equation (7), the upper bound of the guards will be $\lfloor 7n/15 \rfloor$ which is smaller than $\lfloor n/2 \rfloor$. So, the expected number of the guards equals the lower bound of the required guards.

It is important to note that in this analysis, we have considered only the second-order neighbors and by considering other levels of non-adjacent neighbors it is possible to obtain better results.

Also, we can use the following preprocess to improve the efficiency of the algorithm and reduce the number of the selected guards. Before starting the algorithm and as preprocess, the list of all visible faces for each vertex is prepared and during the execution of the algorithm, when a vertex is selected as a guard, all of the visible faces from its list will be marked as covered. Then, as the algorithm proceeds, when a new face appears at the boundary of the TIN, it will be checked to see if it has been marked before. If so, it does not require any more guard and it will be removed from the TIN. To use this improvement it should be paid more preprocessing costs that will increase the running time of the algorithm by $O(n^3)$ to produce the list of the visible faces of each vertex.

4 Concluding Remarks

In this paper we presented a new algorithm for guarding TINs which includes two phases: first, the initial TIN is reduced to a collection of simple polygons. This is done by removing some vertices of the TIN and assigning a guard to

any one of them. Then, by using an existing algorithm the resulting polygons are covered.

The running time of the presented algorithm is $O(n)$ in which n is the number of the vertices of the TIN. The upper bound of the number of the assigned guards is $\lfloor 2n/3 \rfloor$ in worst case and $\lfloor n/2 \rfloor$ in average. In addition, a method was proposed to improve the efficiency of the algorithm by paying more costs as preprocess.

One of the positive aspects of this algorithm is that, during the process of guard assignment, instead of just considering the local visibility, it is possible to use the global visibility of the previously selected guards to cover some non-adjacent faces. This will reduce the number of the guards especially on real surfaces as verified by experimentation.

References:

[1] J. Urrutia, Art gallery and illumination problems, *in: J.-R. Sack, J. Urrutia (Eds.), Handbook on Computational Geometry, Elsevier Science*, Amsterdam, 2000.

[2] R. Cole, M. Sharir, Visibility Problems for Polyhedral Terrains, *Journal of Symbolic Computation* Vol. 7, pp. 11-10, 1981.

[3] S. Eidenbenz, Approximation algorithms for Terrain Guarding, *Information Processing Letters (IPL)*, Vol. 82, pp. 99-105, 2002.

[4] Ana Paula Tomas, Antonio Leslie Bajuelos, Fabio Marques, Approximation Algorithms to Minimum Vertex Cover Problems on Polygons and Terrains, *In P.M.A Sloot et al. (eds.), Proceedings of the International Conference on Computational Science (ICCS 2003)*, Lecture Notes in Computer Science 2657, Springer-Verlag, pp. 869-878, 2003.

[5] S. Eidenbenz, (In-)Approximability of visibility problems on polygons and terrains, *PhD thesis*, Diss. ETH No. 13683, ETH, Zurich, 2000.

[6] P. Bose, T. Shermer, G. Toussaint, B. Zhu, Guarding polyhedral terrains, *Computational Geometry-Theory and Applications*, Vol. 7:3:, pp. 173-185, 1997.

[7] P. Bose, D. Kirkpatrick, Z. Li, Efficient algorithms for guarding or illuminating the surface of a polyhedral terrain, *Proc. of the Eight Canadian Conference on Computational Geometry*, 1996.

[8] B. Chazelle, Triangulating a simple polygon in linear time, *Discrete and Computational Geometry*, Vol. 6, pp. 485-524, 1991.

[9] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry Algorithms and Applications, *Springer*, pp. 45-60, 1997.