Simplifying Planar Visibility Polygons

Alireza Zarei and Mohammad Ghodsi

[1] Computer Engineering Department
Sharif University of Thechnology
[2] IPM School of Computer Science

Abstract. Boundary of a region illuminated by a light source may be composed of many vertices and points. Because of the limited resolution of the display screens, there is no need to exactly maintain all the detail of the boundaries. Instead, an approximation of the boundary is always sufficient in realistic applications. This is also true for the visible region of an observer. The classical line simplification methods can be used to approximate such boundaries by simpler polygons. Unfortunately, the error function used in these methods does not consider the position of the observer (light source) which is an important parameter in our simplification. In this paper, we propose an error function that considers the distance of the boundary polygon and the observer and describe an efficient method for its computation. We then illustrate that this error function can be embedded in current offline simplification algorithms without increasing the time and space complexities. Moreover, we propose a method to simplify such regions in a streaming data model in which the boundary vertices are given as a stream of points and we do not have enough memory to maintain the whole data. Our method uses $O(\frac{k^2}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(\frac{k}{\sqrt{\epsilon}\log\epsilon})$ amortized time. In this method, the error of the resulting simplification with $2k$ points is not bigger than $(2+\epsilon)$ times the error of the optimal solution with k points in which we can store all points. This method can also be used for point, segment or polygon observers inside planer scenes.

Key words: Computational geometry, planar visibility, line simplification

1 Introduction

In many applications of computer graphics and robotics, we are asked to compute the region visible from an observer (or a light source), or the set of illuminated points. This is a basic problem in computational geometry and there are many solutions for its different versions [16–22].

In a planer scene which is composed of a set of polygonal objects in the plane, two points are visible from each other if their connecting segment does not intersect the scene objects. The set of points visible from a point q is called its visibility polygon and is denoted by $VP(q)$. The visibility polygon of a point in a planer domain is always a star-shaped simple polygon.

In real applications, an observer usually has a limited vision power, *i.e.*, it can not distinguish small visibility differences at far distances. Moreover, the required space to maintain the exact visibility polygon is too high and it will be impossible to maintain such polygons exactly. On the other hand, the accuracy of the display screens is also limited. That is, to display such a polygon on a display screen, only its approximation is displayed.

The boundary of a visibility polygon, simply referred to by visibility polygon in the rest of this paper, is composed of many consecutive line segments, some of which may be so far from the observer. The observer in above applications may be unable to distinguish between the segments endpoints. As a result, it will see several segments as a single line segment. Figure 1

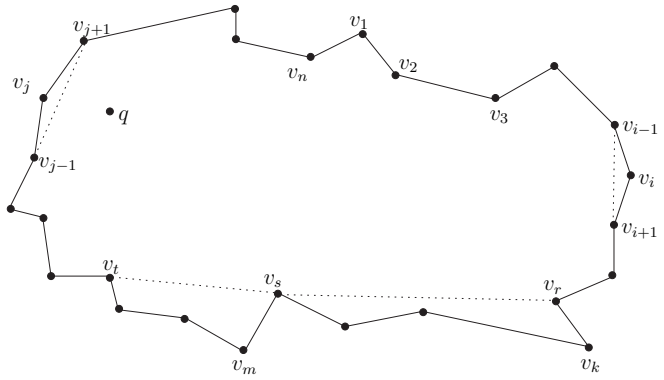

Fig. 1. In simplifying $VP(q)$, the vertex v_i which is farther then v_j from the observer q is a better candidate for elimination.

depicts such a condition in which the observer q sees segments $v_1v_2, v_2v_3, \ldots, v_nv_1$ but segments $v_{i-1}v_i$ and v_iv_{i+1} are farther from q than segments $v_{j-1}v_j$ and v_jv_{j+1}. Therefore, q can see the correct position of point v_j while the point v_i is seen as a point of the segment $v_{i-1}v_{i+1}$.

In this paper, we clarify our motivation and consider the problem of simplifying (approximating) the visibility polygon of such observers inside a polygonal domain.

This problem is a special case of the classical line simplification problem for which there are several algorithms. These methods approximate a given path of line segments by another path with smaller number of segments which minimizes the difference between the initial and the simplified paths. This difference, to be formally defined later, is called the *error* of this simplification.

There are two optimization goals in the line simplification algorithms: min-k and min-δ. In the min-k version, there is a given error threshold and we are to use the minimum number of vertices in the simplified path meeting the error threshold. In min-δ, we are allowed to use at most k vertices for some given k in the simplified path and the goal is to minimize the error of the simplification.

There are many variants of the line simplification problem. In its *restricted* version, the vertices of the simplified path are required to be selected from the set of the vertices of the initial path. Some results on the *unrestricted* version can be found in [9–11, 13]. For the *restricted* version, the main algorithms can be found in [7, 12, 14, 6, 15, 2, 4, 8]. Also, an efficient approximation algorithm can be found in [2] which approximately solves the *restricted* version of this problem.

These simplification methods use different error functions to compute the difference between the initial and the simplified paths. Almost all of the referenced algorithms solve the line simplification problem under the Hausdorff distance for L_1, L_2 or L_∞ metrics. Line simplification under the Fréchet distance has been considered only by Godau and Alt [8, 4].

Unfortunately, none of the error functions used in these algorithms is proper for our purpose of simplifying visibility polygons. In our target applications, the vertices of the path that are closer to the observer are more important than the farther points. Precisely, we need an approximating error function which considers the distance between the points of the visibility polygon and the observer.

To solve this problem, we define a proper approximating error function which considers this distance. We prove that this error function can be computed efficiently and can be used along with current simplification methods without increasing their time or space complexities. Therefore, our target problem can be solved efficiently under min-k or min-δ optimization goals.

We further consider the cases in which the observer is like a radar inside a dynamic environment that circularly sweeps its neighbor and draws its visibility polygon. In such applications, the visible points are given continuously as a stream of input data and we assume that it is impossible to maintain and show all of these points. Therefore, it is necessary to approximate the exact visibility polygon by another polygon of smaller number of vertices.

In this model, regardless of the number of the points in the input path, we must simplify the path by at most k points. Also, we must continuously update the simplification as new points are received. For this version of the problem, our proposed method uses $O(\frac{k^2}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(\frac{k}{\sqrt{\epsilon}\log\epsilon})$ amortized time. Then, the error of the resulting simplification with $2k$ points is not bigger than $(2+\epsilon)$ times the error of the optimal simplification with k points. This method is based on the general algorithm proposed in [1].

Our error function can be extended for different observer types like segment observers which is done in this paper. To the best of our knowledge, the results of this paper are the first in this area and there are several interesting open directions in applying and extending this notion.

The rest of this paper is organized as follows: in Section 2, our *visibility-dependent* simplification error function is described and we use this metric to simplify visibility polygons in non-stream input models. In Section 3, we solve the problem in streaming data model. The extensions and future works of the proposed method are given is Section 4 and the paper is summarized and concluded in Section 5.

2 Visibility-Dependent Simplification

We focus on the restricted version of the line simplification problem. For this problem, let P be a path defined by a sequence of points $p_0, p_1, p_2, \ldots, p_n$. Any subsequence $Q = q_0, q_1, \ldots, q_l, q_{l+1}$ of P is a l-simplification of P if $q_0 = p_0$ and $q_{l+1} = p_n$. In this simplification, any segment q_iq_{i+1} of Q $(0 \leq i \leq l)$ is the corresponding simplification of the subpath $p_s, p_{s+1}, \ldots, p_t$ of P where $q_i = p_s$ and $q_{i+1} = p_t$. In other words, we have replaced the subpath $p_s, p_{s+1}, \ldots, p_t$ of P with segment q_iq_{i+1} in Q.

Therefore, Q is an approximation of P and can be stored using smaller size of memory, however, at the cost of losing the accuracy of P. Assume that *error* is our error function used to compare similarity of Q and P. Using this metric, we denote the error of this approximation by $error(Q)$ and it is defined to be the maximum error of segments $q_iq_{i+1}(0 \leq i \leq l)$ under this metric. The error of a segment q_iq_{i+1} under a metric *error* is denoted by $error(q_iq_{i+1})$ and is defined to be the error of approximating the subpath $p_s, p_{s+1}, \ldots, p_t$ by segment q_iq_{i+1} under this error metric. Usually, the definition of the error metric *error* depends on the application.

Hausdorff error function, $error_h$, is the metric used in almost all simplification algorithms. For a segment q_iq_{i+1} which is the simplification of a subpath $p_s, p_{s+1}, \ldots, p_t$, $error_h(q_iq_{i+1})$ is defined as the maximum euclidian distance of the points $p_s, p_{s+1}, \ldots, p_t$ from segment q_iq_{i+1}. For example, $error_h(p_rp_s)$ in Figure 1 is $|p_kp_r|$ which is the distance of the point p_k from segment p_rp_s. Also, in this figure, $error_h(p_sp_t) = d(p_m, p_sp_t)$ where $d(p_m, p_sp_t)$ is the distance of the point p_m from segment p_sp_t.

The Hausdorff error function only depends on the initial and the simplified paths and therefore, is not proper for simplifying visibility polygons in which the position of the observer has an important role. Assume that $P = p_1, p_2, \ldots, p_n, p_1$ of Figure 1 is the visibility polygon of a point observer q. Here, p_j is closer to the observer than p_i which is assumed to be too far from q. If we are to simplify P by removing one point and we have only two choices p_i and p_j, it would be better to remove p_i while if we use Hausdorff error function,

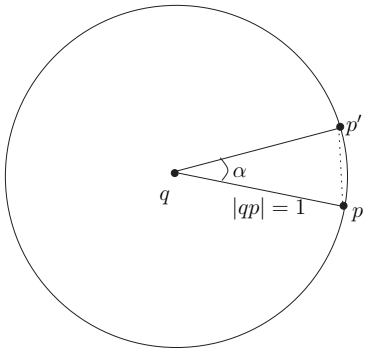

Fig. 2. Vision distinction power of a point observer.

p_i will be removed. In order to use current simplification algorithms, we must formalize this issue as an error function to be used in these algorithms.

For this purpose, we define the *vision distinction power* of a point observer q as the maximum distance of two points, which are at unit distance from q, such that q can distinguish between them as two separate points. The vision distinction power of a point observer q, denoted by $vdp(q)$, means that if the distance between two points that lie on the unit circle centered at q is less than $vdp(q)$, q will see them as a single point. As shown in Figure 2, if $vdp(q)$ is too small and $|pp'| = vdp(q)$, we can say that $vdp(q) = |pp'| \simeq |qp|.\alpha \simeq \alpha$.

Therefore, we can say that the vision distinction power of a point observer q is the smallest angle of two rays emitted from q such that q can see different points along these rays. For small values of α, a point observer q of $vdp(q) = \alpha$ sees all points of the disk centered at o of diameter d as a single point if $d \leq \alpha|qo|$. The reason is that this disc is contained between two rays from q which their angle is less than α and according to the definition of vdp, these points are seen as a single point.

Now, we are ready to define our simplification error function which considers vdp. As shown in part A of Figure 3, assume that we are to approximate the path p_ipp_j, a part of the visibility polygon of the observer q, by segment p_ip_j. From the viewpoint of q, this approximation maps the point p to point p'. Also, other points of segments p_ip and pp_j are mapped to their corresponding points of segments p_ip' and $p'p_j$.

The corresponding visibility-dependent error of this simplification for a point t on the path p_ipp_j is denoted by $error_{vis}(t, p_ip_j)$ and is defined as $\frac{|tt'|}{|tq|}$ where t' is the intersection point of segments tq and p_ip_j. This means that in this simplification and at distance $|tq|$ from the observer we have violated from the initial path by a value of $|tt'|$. This definition is also extended to paths of more internal vertices. The visibility polygon of q is a star-shaped polygon and p lies between p_i and p_j on the boundary of this polygon. Therefore, the supporting line of pq always intersects p_ip_j. If p_i, p_j and q are collinear, p and all other points of the polygon boundary from p_i to p_j must also lie on segment p_ip_j. For such situations, the error of all points of the path from p_i to p_j is zero which corresponds to our definition of the error function.

In some cases like part B of Figure 3, tq does not intersect p_ip_j. For such situations, point t is mapped to point t' which is the intersection point of p_ip_j and the supporting line of tq. In these cases, the visibility-dependent error of point t is defined to be $\frac{|tt'|}{|t'q|}$. Comparing the definition of $error_{vis}$ function for these two cases, when the corresponding values of $|tq|$ in parts A and B of Figure 3 are equal, we assign greater error value to point t in part A. This means that in the same situations we prefer to simplify using the outer diameters of the visibility polygon compare to the internal ones. Another benefit of this definition is that the error function will be monotone to be defined and used later.

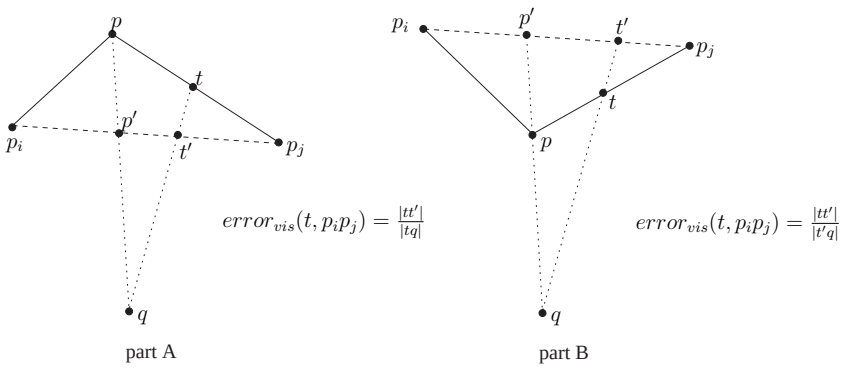

Fig. 3. Visibility-dependent simplification error.

Our visibility-dependent error function associated with a path $p_i, p_{i+1}, \ldots, p_j$ simplified by segment p_ip_j, denoted by $error_{vis}(p_ip_j)$, is defined to be the maximum visibility-dependent error of points of this path.

This definition for visibility-dependent error function strongly relates to the notion of width. The width of a set of points with respect to a given direction $\overrightarrow{d}$ is the minimum distance of two lines being parallel to $\overrightarrow{d}$ that enclose the point set. Let $P_L(i,j)(P_U(i,j))$ be the set of points of subpath $P(i,j) = p_i, p_{i+1}, \ldots, p_j$ that lie in the closed half plane defined by the supporting line of p_ip_j which contains(does not contains) the point observer q. We denote by $w_L(i,j)(w_U(i,j))$ the width of the points of $P_L(i,j)(P_U(i,j))$ with respect to the direction $\overrightarrow{p_ip_j}$. We have,

Lemma 1. *For a subpath $P(i,j) = p_i, p_{i+1}, \ldots, p_j$ of $VP(q)$,*

$$error_{vis}(p_ip_j) = \max\left(\frac{w_U(i,j)}{d(q,p_ip_j)+w_U(i,j)}, \frac{w_L(i,j)}{d(q,p_ip_j)}\right)$$

where $d(q, p_ip_j)$ is the orthogonal distance of point q from the supporting line of p_ip_j.

Proof. From Thales theorem, for any point p on path $P(i,j)$ that lies in opposite side of p_ip_j relative to q we have $error_{vis}(p, p_ip_j) = \frac{d(p,p_ip_j)}{d(p,p_ip_j)+d(q,p_ip_j)}$. Therefore, the maximum error of these points is $\frac{w_U(i,j)}{d(q,p_ip_j)+w_U(i,j)}$. Similarly, for a point p on path $P(i,j)$ that lies in the same side of p_ip_j relative to q we have $error_{vis}(p, p_ip_j) = \frac{d(p,p_ip_j)}{d(q,p_ip_j)}$ and their maximum is $\frac{w_L(i,j)}{d(q,p_ip_j)}$. □

A direct consequence of this lemma is that the associated error of a segment p_ip_j belongs to a vertex $p_k(i \le k \le j)$ which makes computation of this error function straightforward. Using this result we can simply compute the corresponding error of any segment p_ip_j that may be appeared in simplification during the simplification process by only checking vertices of the subpath $P(i,j)$.

Fortunately, algorithms proposed for both restricted and unrestricted versions of the line simplification problem do not require any special property for the error function and we can plugged our error function into. Moreover, this error function can be used under $\min -k$ and $\min -\delta$ optimization goals as well. The only change in these algorithms is to use our error function for a segment p_ip_j when we want to simplify the path $p_i, p_{i+1}, \ldots, p_j$ with this segment.

As an example, we use our error function in Imai and Iri[15] approach for solving the $\min -k$ version of the line simplification problem. In this approach we build a directed acyclic graph G over the vertices of the path $P = p_0, p_1, \vdots p_n$. For any $p_i \in P$, there is a vertex p_i in

G and an edge p_ip_j is added to this graph if $error_{vis}(p_ip_j)$ is not greater than the allowed error threshold. The shortest path from p_0 to p_n in G is the solution of this version of the simplifying P. As seen here, the only change is to use our visibility-dependent error function whenever we want to compute the corresponding error of a segment p_ip_j.

3 Visibility-Dependent Simplification in Streaming Model

In Section 2, we described a method to measure the associated error of simplifying the visibility polygon of a point observer in which the position of the observer is considered. Also, we showed how this error function can be used in current offline simplification algorithms. These algorithms are only applicable to situations where all of the points of the path are available in memory. In some applications we can not maintain the whole path because of the limited amount of memory or unnecessity of maintaining these points. For example, consider a radial sweep line which trace the scene around a point observer. In such applications, we want to compute an approximation of the visibility polygon as the visible points are identified by the sweep line.

Formally, the vertices of the visibility polygon are given as a stream of input data and we want to simplify the path. Abam *et.al* proposed a general algorithm that can be used to simplify a path which its vertices are given as a stream of input points[1]. The min $-k$ version of the line simplification is not well-defined in streaming model. Thus, their algorithm only solves the min $-\delta$ version of the line simplification problem.

In order to use this algorithm on a path $P(n) = p_0, p_1, \ldots, p_n$ with an error function *error*, two conditions must be satisfied:

- *error* must be a *c-monotone* error function on the path $P(n)$ for any $n > 0$. This means that for any two segments p_ip_j and p_lp_m such that $i \leq l \leq m \leq j$ and p_i, p_j, p_l and p_m are vertices of the path $P(n)$, we have
$$error(p_lp_m) \leq c.error(p_ip_j).$$
In other words, an error function is c-monotone if the error of a segment can not be worse than c times the error of any segment that encloses it.
- There must be an *e-approximate* error oracle for *error* on the path $P(n)$ to be defined as follows. In streaming models, we may lose some vertices of the subpath $P(i, j)$ between points p_i and p_j. Then, we can not compute the exact value of the error function for this segment and we must approximate this error value. We denote the approximated error value of a segment p_ip_j by $error^*(p_ip_j)$. We call the procedure that computes this approximation as our error *oracle*. An error oracle is e-approximate if for any segment p_ip_j for which the oracle is called by the algorithm we have
$$error(p_ip_j) \leq error^*(p_ip_j) \leq e.error(p_ip_j).$$

Having these two conditions, the algorithm of Abam *et al.* [1] simplifies a streaming path P by a path Q of at most k internal vertices. The time the algorithm needs to update the simplification upon the arrival of a new point is $O(\log k)$ plus the time spent by the error oracle. Besides the storage needed for the simplification Q, the algorithm needs $O(k)$ storage plus the storage needed by the error oracle. The algorithm is quite simple (assume that $k = l$):

Suppose we have already handled the points $p_0, \ldots, p_n$. (We assume $n > l + 1$; until that moment we can simply use all points and have zero error.) Let $Q := q_0, q_1, \ldots, q_l, q_{l+1}$ be the current simplification. The algorithm will maintain a priority queue $\mathcal{Q}$ that stores the points q_i with $1 \leq i \leq l$, where the priority of a point is the error (as computed by the oracle) of the link $q_{i-1}q_{i+1}$. In other words, the priority of q_i is (an approximation of) the error that is incurred when q_i is removed from the simplification. Now the next point p_{n+1} is handled as follows:

1. Set $q_{l+2} := p_{n+1}$, thus obtaining an $(l+1)$-simplification of $P(n+1)$.
2. Compute $error^*(q_l q_{l+2})$ and insert q_{l+1} into $\mathcal{Q}$ with this error as priority.
3. Extract the point q_s with minimum priority from $\mathcal{Q}$; remove q_s from the simplification.
4. Update the priorities of q_{s-1} and q_{s+1} in $\mathcal{Q}$.

The error of the simplification Q obtained by this algorithm for $l = 2k$ is at most ce times the error of the optimal simplification of P with k points in non-streaming model which we have all points in memory. So, in order to use this algorithm we must show that our visibility-dependent error function, $error_{vis}$, is c-monotone and we must propose an error oracle to approximate the error of any segment $p_i p_j$ for which the oracle is called in this algorithm.

Lemma 2. *Over the visibility polygon of a point observer, the visibility-dependent error function $error_{vis}$ is 2-monotone.*

Proof. Assume that points p_i, p_j, p_l and p_m lie on $VP(q)$ such that $i \leq l \leq m \leq j$ and $error_{vis}(p_l p_m)$ belongs to a point p_k where $l \leq k \leq m$ and p'_k and p''_k are respectively the intersection points of the supporting line of qp_k and segments $p_l p_m$ and $p_i p_j$. $VP(q)$ is a star-shaped polygon and q is a point in its center. Following the order of points on the boundary of this polygon, the supporting line of segments $p_l q$, $p_m q$ and $p_k q$ intersect segment $p_i p_j$ and the supporting line of $p_k q$ intersects $p_l p_m$. Therefore, there are six permutations for positions of points p_k, p'_k and p''_k on the supporting line of qp_k (shown in Figure 4). For all of these configurations we have

$$error_{vis}(p_i p_j) \geq \max(error_{vis}(p_l, p_i p_j), error_{vis}(p_m, p_i p_j), error_{vis}(p_k, p_i p_j)),$$

and

$$error_{vis}(p'_k, p_i p_j) \leq \max(error_{vis}(p_l, p_i p_j), error_{vis}(p_m, p_i p_j)).$$

Consequently, we have

$$error_{vis}(p_i p_j) \geq \max(error_{vis}(p'_k, p_i p_j), error_{vis}(p_k, p_i p_j)).$$

We prove the lemma for all these configuration by showing that

$$error_{vis}(p_l p_m) = error_{vis}(p_k, p_l p_m) \leq 2\max(error_{vis}(p'_k, p_i p_j), error_{vis}(p_k, p_i p_j)) \leq$$
$$2error_{vis}(p_i p_j).$$

The first equality is our assumption that p_k has the maximum error on $p_l p_m$ among all points of path $p_l, p_{l+1}, \ldots, p_m$ and we have already shown the last inequality. Therefore, it is only enough to show the middle inequality.

- Case 1(shown in part A of Figure 4): In this configuration we have,

$$error_{vis}(p_k, p_l p_m) = \frac{|p_k p'_k|}{|p_k q|} \leq \frac{|p_k p''_k|}{|p_k q|} = error_{vis}(p_k, p_i p_j) \leq$$
$$2\max(error_{vis}(p'_k, p_i p_j), error_{vis}(p_k, p_i p_j)).$$

- Case 2(shown in part B of Figure 4): Here, if $|p_k p''_k| \geq |p''_k p'_k|$ then we have,

$$error_{vis}(p_k, p_l p_m) = \frac{|p_k p'_k|}{|p_k q|} = \frac{|p_k p''_k| + |p'' k p'_k|}{|p_k q|} \leq \frac{2|p_k p''_k|}{|p_k q|} \leq$$
$$2error_{vis}(p_k, p_i p_j) \leq 2\max(error_{vis}(p'_k, p_i p_j), error_{vis}(p_k, p_i p_j)),$$

and if $|p_k p''_k| < |p''_k p'_k|$ then we have,

$$error_{vis}(p_k, p_l p_m) = \frac{|p_k p'_k|}{|p_k q|} \leq \frac{2|p'' k p'_k|}{|p_k q|} \leq \frac{2|p''_k p'_k|}{|p''_k q|} \leq$$
$$2error_{vis}(p'_k, p_i p_j) \leq 2\max(error_{vis}(p'_k, p_i p_j), error_{vis}(p_k, p_i p_j)).$$

So in both conditions, the lemma is valid in this case.

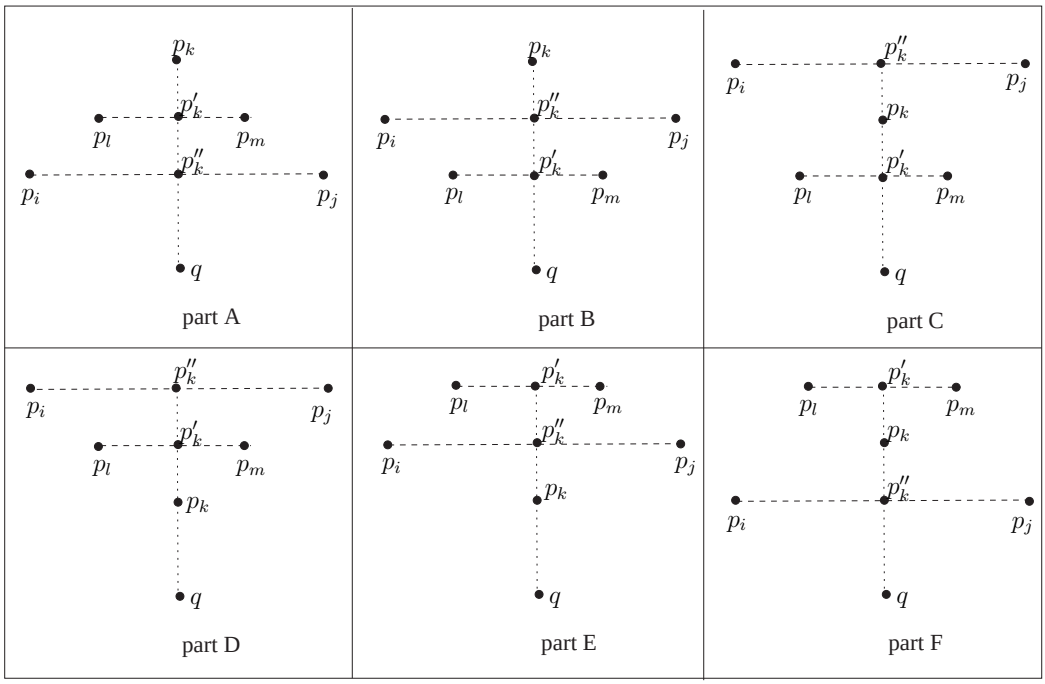

Fig. 4. The visibility-dependent error function is 2-monotone.

– Case 3(shown in part C of Figure 4): For this case, assume that $|p_k p''_k| = x|p_k q| = x(|p_k p'_k| + |p'_k q|)$. Then,

$$\max(error_{vis}(p'_k, p_i p_j), error_{vis}(p_k, p_i p_j)) \geq error_{vis}(p'_k, p_i p_j) = \frac{|p_k p'_k| + |p_k p''_k|}{|p_k p''_k| + |p_k q|} =$$

$$\frac{|p_k p'_k| + x(|p_k p'_k| + |p'_k q|)}{x|p_k q| + |p_k q|} = \frac{(x+1)|p_k p'_k| + x|p'_k q|}{(x+1)|p_k q|} = error_{vis}(p_k, p_l p_m) + \frac{x|p'_k q|}{(x+1)|p_k q|} \geq error_{vis}(p_k, p_l p_m).$$

– Case 4(shown in part D of Figure 4): The proof of this case is exactly the same as case 3.
– Case 5(shown in part E of Figure 4): The proof of this case is exactly the same as case 2.
– Case 6(shown in part F of Figure 4): The proof of this case is exactly the same as case 1.

So, we proved that in all cases, $error_{vis}(p_l p_m) \leq 2error_{vis}(p_i p_j)$. Also, it is simple to show that this upper bond is tight in cases 2 and 5. □

Now, we propose an approximating procedure that approximates $error_{vis}(p_i p_j)$, the error value of any segment $p_i p_j$ for which the simplification algorithm is called.

According to Lemma 1, the approximating oracle can approximate $d(q, p_i p_j)$, $w_L(i, j)$ and $w_U(i, j)$ to find an approximation of $error_{vis}(p_i p_j)$. It is easy to find the exact value of $d(q, p_i p_j)$. We use the method described by Agarwal and Yu [3] to approximate w_L and w_U.

Agarwal and Yu [3] have described a streaming algorithm for maintaining a core-set that can be used to approximate the width of a set of points in any direction. Their algorithm requires $O(\frac{1}{\sqrt{\epsilon}})$ space and $O(\frac{1}{\log \epsilon})$ amortized time per point to maintain a core-set from which the width of the input stream can be computed efficiently. This is done by additionally maintaining the convex hull of the core-set using the data structure by Brodal and Jacob [5]. This data structure uses linear space and can be updated in logarithmic time. Also it supports queries for the extreme point in a given direction in logarithmic time. Using these results,

we have an $(1+\epsilon)$-approximate error oracle for $error_{vis}$ and the value of $error_{vis}(p_ip_j)$ can be computed in $O(\frac{1}{\log \epsilon})$ time.

Lemma 3. *There is a $(1+e)$-approximate error oracle for the visibility-dependent error function on visibility polygon of a point observer that uses $O(\frac{k^2}{\sqrt{\epsilon}})$ storage and has $O(\frac{k}{\sqrt{\epsilon}\log \epsilon})$ amortized update time where k is the number of the internal points of the simplification.*

Proof. Assume that $Q = q_0, q_1, \ldots, q_k, q_{k+1}$ is the current simplification of path $P(n) = p_0, p_1, \ldots, p_n$. Any one of the segments q_iq_j where $0 \leq i \leq j \leq k+1$, may appear in the simplification in future and we must be able to approximate their $error_{vis}(q_iq_j)$. Thus, we maintain two core-set structures for each segment q_iq_j. One of these core-sets is used to approximate $w_U(i,j)$ and only the points of $P_U(i,j)$ are added to this core-set. The other core-set is maintained for approximating $w_L(i,j)$ and only contains the points of $p_L(i,j)$.

Therefore, we maintain $O(k^2)$ core-sets of size $O(\frac{1}{\sqrt{\epsilon}})$. Considering a new point $p_{n+1} = q_{k+2}$, we must create $O(k)$ new core-sets, one for each of the lines q_ip_{n+1} for $0 \leq i \leq k+1$. These core-sets are created by copying the corresponding core-sets of q_ip_{k+1} and inserting point p_{n+1} into them using the algorithm by Agarwal and Yu [3]. When a point q_s is removed form the simplification, the core-sets of all segments that start or end at q_s have become meaningless and are therefore deleted.

In total, $O(\frac{k^2}{\sqrt{\epsilon}})$ storage is needed for the $O(k^2)$ core-sets. The update of the oracle involves creation of $O(k)$ core-sets from the current ones. This must be done by comparing the position of the points of the current core-sets to the new segment for which the core-sets are created and inserting the points in proper(upper or lower) core-set of the new segment. So, we need $O(k\frac{1}{\sqrt{\epsilon}}\frac{1}{\log \epsilon})$ amortized time to create the new core-sets. The new point is added to these core-sets in $O(\frac{k}{\log \epsilon})$ amortized time. Therefore, the time the oracle needs to process a new point is $O(\frac{k}{\sqrt{\epsilon}\log \epsilon})$. □

Combining the result of lemmas 2, 1 and 3 with the algorithm of Abam *et al.* [1] described at the beginning of this section, we have the following result on simplifying the visibility polygon of a point observer based on the visibility-dependent error function:

Theorem 1. *There is a streaming algorithm that maintains a $2k$-simplification for $VP(q)$ under the visibility-dependent error function. This algorithm uses $O(\frac{k^2}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(\frac{k}{\sqrt{\epsilon}\log \epsilon})$ amortized time and the error of the result simplification is not larger than $(2+\epsilon)$ times the error of the optimal offline k-simplification.*

4 Extensions and Future Works

In previous sections, we defined the visibility-dependent error function only for point observers. This definition can be extended to other kinds of observers. Assume that our observer is a line segment. There are two types of visibility polygons for a segment: weak and strong. Weak visibility polygon of segment pq is the set of points that are visible from at least one point of pq. Strong visibility polygon is defined to be the set of points that are visible from all points of pq.

Assume that we want to approximate a subpath $p_ip_{i+1}, \ldots, p_j$ of the weak visibility polygon of a segment observer pq by the segment p_ip_j and p_x is a point on this path. A reasonable definition of visibility-dependent error function of this approximation for p_x is $\min(error_{vis}(p_x, p_ip_j)$ with respect to point $p'_x)$ over all points p'_x on segment pq where p_x and p'_x are visible from each other. A reasonable visibility-dependent error function for strong visibility polygon is to use max instead of min in the above definition.

These definitions can also be extended to simple polygon observers. Applying the proposed algorithm for simplifying the visibility polygon of these types of observers is an interesting future work that may follow this work.

Another practical and useful extension of this paper is to apply the notion of simplifying visibility polygon for a moving observer. For moving observers we need online simplification algorithms instead of offline or streaming methods used in this paper.

Simplifying the visible region of an observer in 3D has many applications as well as the theoretic interests. So, another interesting problem is to extend this definition of error function to higher dimensions

Finally, implementing this algorithm and comparing with the image drawn on limited resolution display screens is a good method to find the improvement points of this work.

5 Conclusions

In this paper, we considered the problem of simplifying the visibility polygon of an observer inside a planer scene. This problem have many applications in computer graphics, games, robotics, path planning and GIS. We first defined a visibility-dependent error function to compare different simplifications, formally. For our definition of the visibility-dependent simplification, we described how to use current simplification methods to simplify a visibility polygon, efficiently.

Then, we proposed a simplification method for conditions where the points of the visibility polygon are given as a stream of points and we do not have enough storage to maintain all points. Here, our method uses $O(\frac{k^2}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(\frac{k}{\sqrt{\epsilon}\log\epsilon})$ amortized time. In this method, the error of the resulting simplification with $2k$ points is not bigger than $(2+\epsilon)$ times the error of the optimal simplification with k points in which we can store all points.

Also, we described how to extend our visibility-dependent error function for other observer types and several interesting extension points and future works are introduced in Section 4. This is the first attempt in defining formal simplification criteria which considers the visibility properties and it can be used in real applications in which the exact boundaries are not displayed or maintained.

References

1. M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei. Streaming Algorithms for Line Simplification. *23rd ACM Symp. on Computational Geometry (SoCG)*, pages 175–183, 2007.
2. P.K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete & Compututationl Geometry* 23(2):273–291, 2000.
3. P.K. Agarwal, H. Yu. A Space-Optimal Data-Stream Algorithm for Coresets in the Plane. In: *Proc. 23th ACM Symposium on Computational Geometry (SOCG)*, pages 1–10, 2007.
4. H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal on Computational Geometry and Applications* 5:75–91 1995.
5. G.S. Brodal and R. Jacob. Dynamic Planar Convex Hull. In *Proc. 43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 617–626, 2002
6. W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *Proc. 3rd Annual International Symposium on Algorithms and Computing (ISAAC)*, LNCS 650, pages 378–387, 1992.
7. D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer* 10:112–122, 1973.
8. M. Godau. A natural metric for curves: Computing the distance for polygonal chains and approximation algorithms. In *Proc. 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 127–136, 1991.

9. M.T. Goodrich. Efficient piecewise-linear function approximation using the uniform metric. *Discrete & Computational Geometry* 14:445–462, 1995.
10. L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *International Journal of Computatioanl Geometry and Applications* 3:383–415, 1993.
11. S.L. Hakimi and E.F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models Image Processing* 53:132–136, 1991.
12. J. Hershberger and J. Snoeyink. An $O(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification. In *Proc. 10th ACM Symposium on Computational Geometry (SOCG)*, pages 383–384, 1994.
13. H. Imai and M. Iri. An optimal algorithm for approximating a piecewise linear function. *Journal of Information Processing* 9(3):159–162, 1986.
14. H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. In: G.T. Toussaint (ed.), *Computational Morphology*, North-Holland, pages 71–86, 1988.
15. A. Melkman and J. ORourke. On polygonal chain approximation. In: G.T. Toussaint (ed.), *Computational Morphology*, North-Holland, pages 87–95, 1988.
16. H. Gindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2:186197, 1981.
17. P. J. Heffernan and J. S. B. Mitchell. An optimal algorithm for computing visibility in the plane. *SIAM Journal of Computing*, 24(1):184201, 1995.
18. S. Suri and J. ORourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proc. of the second annual symposium on Computational geometry*, pages 1423, 1986.
19. M. Pocchiola and G. Vegter. The visibility complex. *Internattional Journal of Computation Geometry and Applications*, 6(3):279308, 1996.
20. A. Zarei and M. Ghodsi. Efficient computation of query point visibility in polygons with holes. In *Proc. 21st Annual Symposium on Computational Geometry*, pages 314320. ACM, 2005.
21. M. T. B. Aronov, L Guibas and L. Zhang. Visibility queries and maintenance in simple polygons. *Discrete and Computational Geometry*, 27(4):461483, 2002.
22. A. L. P. Bose and J. I. Munro. Efficient visibility queries in simple polygons. *Computational Geometry: Theory and Applications*, 23(3):313335, 2002.