# On Planar Visibility Polygon Simplification[*]

Alireza Zarei[†] Mohammad Ghodsi[†]

**Abstract**

The boundary of a region illuminated by a light source may be composed of many vertices and points. In this paper, we propose a criterion to represent such polygons by a smaller number of vertices and, show how this criterion can be used in offline and streaming models.

## 1 Introduction

In a planar 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 planar domain is always a star-shaped simple polygon. 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.

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.

In this paper, we consider the problem of simplifying (approximating) the visibility polygon of such observers inside a polygonal domain. This problem is a special case of the well-known 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-$\delta$. 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-$\delta$, 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.

Almost all current simplification algorithms solve the line simplification problem under the Hausdorff distance for $L_1$, $L_2$ or $L_\infty$ metrics or under the Fréchet distance which are not 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.

To solve this problem, we define a new approximating error function which considers the distance between the points of the visibility polygon and the observer. 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-$\delta$ optimization goals.

We further consider the streaming 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 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].

There is a similar attempt in rendering based simplification by Buzer [4], however, without considering the observer position. 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.

[*]This research was in part supported by a grant from IPM. (No. CS1386-2-01)

[†]Computer Engineering Department - Sharif University of Thechnology, School of Computer Science - Institute for Studies in Theoretical Physics and Mathematics (IPM), zarei@mehr.sharif.edu, ghodsi@sharif.edu

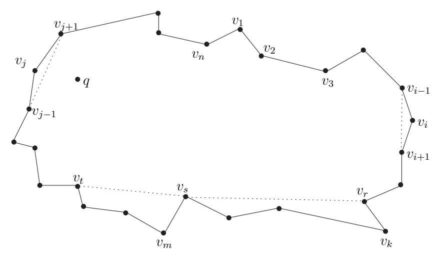
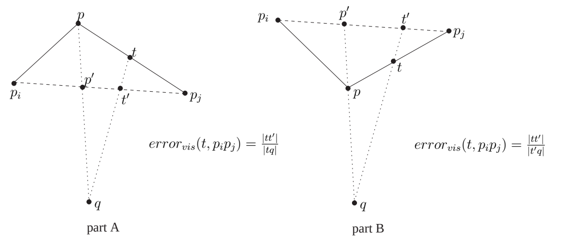


Figure 1: Simplifying *VP*$(q)$.



$p$
$t$
$p'$
$t'$
$p_i$
$p_j$
$error_{vis}(t, p_ip_j) = \frac{|tt'|}{|tq|}$
$q$
part A
$p_i$
$p'$
$t'$
$p_j$
$t$
$p$
$error_{vis}(t, p_ip_j) = \frac{|tt'|}{|t'q|}$
$q$
part B

Figure 2: Visibility-dependent simplification error.

## 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 $q_iq_{i+1}$.

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, $p_i$ will be removed. In order to use current simplification algorithms, we formalize this issue as an error function as follows.

Assume that we are to approximate the path $p_ipp_j$ (See part A of Figure 2), 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 2, $tq$ 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 2 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 this error function is monotone which will be defined and used in Section 3.

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 $VP(q)$,*

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

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

A direct consequence of this lemma is that the associated error of a segment $p_ip_j$ belongs to a vertex $p_k(i \leq k \leq j)$ 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 appear 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 plug 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.

## 3 Visibility-Dependent Simplification in Streaming Model

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 whose vertices are given as a stream of input points[1]. Their algorithm only solves the $\min -\delta$ version of the line simplification problem. Because of the large number of the input vertices, the result of the $\min -k$ version of the line simplification in streaming model, may be too large to store, and therefore, no result exists for.

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, $error(p_lp_m) \leq c.error(p_ip_j)$.
- 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 $2k$ 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 error of the simplification $Q$ obtained by this algorithm 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 $p_ip_j$ 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 $error_{vis}(p_lp_m)$ 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 $p_lp_m$ and $p_ip_j$. $VP(q)$ 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 $p_lq$, $p_mq$ and $p_kq$ intersect segment $p_ip_j$ and the supporting line of $p_kq$ intersects $p_lp_m$. 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 3). For all of these configurations we have

$$error_{vis}(p_ip_j) \geq$$

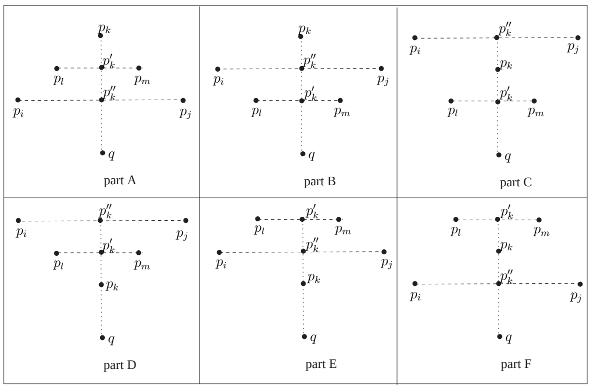


Figure 3: The visibility-dependent error function is 2-monotone.

$\max(error_{vis}(p_l, p_ip_j), error_{vis}(p_m, p_ip_j), error_{vis}(p_k, p_ip_j))$, and

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

Consequently, we have

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

We prove the lemma for all these configuration by showing that

$$\begin{aligned} error_{vis}(p_lp_m) &= error_{vis}(p_k, p_lp_m) \\ &\leq 2\max(error_{vis}(p'_k, p_ip_j), error_{vis}(p_k, p_ip_j)) \\ &\leq 2error_{vis}(p_ip_j). \end{aligned}$$

The first equality is our assumption that $p_k$ has the maximum error on $p_lp_m$ 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. We prove this inequality for the case shown in part A of Figure 3 and skip the other cases. In this configuration we have,

$$error_{vis}(p_k, p_lp_m) = \frac{|p_kp'_k|}{|p_kq|} \leq \frac{|p_kp''_k|}{|p_kq|} = error_{vis}(p_k, p_ip_j)$$
$$\leq 2\max(error_{vis}(p'_k, p_ip_j), error_{vis}(p_k, p_ip_j)).$$

So, we proved that $error_{vis}(p_lp_m) \leq 2error_{vis}(p_ip_j)$. This can be proved for the other cases. Also, it can be shown that this upper bond is tight in cases shown in parts B and E of Figure 3. □

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

According to Lemma 1, the approximating oracle can approximate $d(q, p_ip_j)$, $w_L(i,j)$ and $w_U(i,j)$ to find an approximation of $error_{vis}(p_ip_j)$. It is easy to find the exact value of $d(q, p_ip_j)$. We use the method described by Agarwal and Yu [2] to approximate $w_L$ and $w_U$.

Agarwal and Yu [2] 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 [3]. 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. Therefore, we can prove the following lemma:

**Lemma 3** *There is a $(1+\epsilon)$-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.*

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 4** *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.*

## References

[1] M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei. Streaming Algorithms for Line Simplification. In *23rd ACM Symp. on Computational Geometry (SoCG)*, pages 175–183, 2007.

[2] 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.

[3] 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

[4] L. Buzer. Optimal Simplification of Polygonal Chain for Rendering. In *23rd ACM Symp. on Computational Geometry (SoCG)*, pages 168–174, 2007.