Algorithmica
DOI 10.1007/s00453-011-9488-3

Efficient Observer-Dependent Simplification in Polygonal Domains

Alireza Zarei · Mohammad Ghodsi

Received: 12 October 2009 / Accepted: 4 January 2011

Abstract In this paper, we consider a special version of the well-known line-simplification problem for simplifying the boundary of a region illuminated by a point light source q, or its visibility polygon $VP(q)$. In this simplification approach, we should take the position of q as an essential factor into account to determine the quality of the resulting simplification. For this purpose, we redefine the known distance- and area-distortion error criteria as the main simplification criteria to take into account the distance between the observer q and the boundary of $VP(q)$. Based on this, we propose algorithms for simplifying $VP(q)$. More precisely, we propose simplification algorithms of $O(n^2)$ and $O(n^{4/3+\delta})$ running time for observer-dependent distance-distortion simplification criterion and an $O(n^3)$ simplification algorithm for observer-dependent area-distortion criterion where n is the number of vertices in $VP(q)$. Moreover, we consider the observer-dependent distance-distortion simplification problem in the data streaming model where the vertices of $VP(q)$ are given as a stream and only a constant amount of memory is available.

Keywords Computational geometry · Planar visibility · Line simplification

A. Zarei (✉)
Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran
e-mail: zarei@sharif.edu

M. Ghodsi
Computer Engineering Department, Sharif University of Technology, Tehran, Iran
e-mail: ghodsi@sharif.edu

M. Ghodsi
IPM School of Computer Science, Tehran, Iran

Published online: 14 January 2011

1 Introduction

Motivation In many applications of computer graphics and robotics, we are asked to maintain and process the region visible from an observer, or equivalently, the set of illuminated points from a light source. 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 visible region may be too high and it may be impossible to accurately maintain such boundaries. On the other hand, the resolution of a display screen is always limited and only an approximation of a higher resolution boundary is displayed on such screens. This leads us to the problem of approximating such boundaries by simpler (smaller number of vertices) polygons for faster processing time and smaller storage. As a key factor, this approximation must be done based on the position of the observer; closer parts of the boundary to the observer have greater chance of appearing in the approximation. The reason is easy; in real applications, an observer sees the closer boundaries more clearly whereas it can not distinguish small visibility changes at far distances.

Related Works Our problem is a special case of the classic line simplification problem for which there are several algorithms. These methods approximate a given chain (path) of line segments by another chain of fewer segments in order to minimize the difference between the initial and the simplified chains. This difference, to be formally defined later, is called the *error* of this simplification.

There are two optimization goals in 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 this 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 line simplification problem. In its *restricted* version, the vertices of the simplified path are required to be selected from the set of vertices of the original path. Some results on the *unrestricted* version can be found in [11–13, 16]. For the restricted version, the main algorithms can be found in [3–5, 8–10, 14, 17–19, 21, 23, 24]. Also, an approximation algorithm can be found in [3] which approximately solves the restricted version of this problem.

These simplification algorithms use different criteria to compute the quality of the simplification or the difference between the initial and the simplified paths. Distance-distortion and area-distortion are the most widely used criteria or error functions. In distance-distortion error function, the error of a simplification is defined as the maximum distance between the initial and the simplification paths. This distance is further measured using various metrics including Hausdorff distance for L_1, L_2 or L_∞ metrics [3, 8, 9, 14, 17, 21] and Fréchet distance [4, 10]. In area-distortion, the error of a simplification is defined as a function of the area created between the initial and the simplification paths [5, 18, 19, 23, 24].

Definitions, Notation, and Problem Statement In a planar scene composed of a set of polygonal objects, 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, called its visibility polygon and denoted by $VP(q)$, is always a star-shaped simple polygon in planar domains. In this paper, we interchangeably use visibility polygon for its boundary.

Except for Sect. 3.3, we focus on the restricted version of the line simplification problem. Let P be a path defined by a sequence of points $p_0, p_1, p_2, \ldots, p_n$. A 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 $q_i q_{i+1}$ of Q $(0 \leq i \leq l)$ is the corresponding simplification of subpath $P(s,t) = p_s, p_{s+1}, \ldots, p_t$ of P where $q_i = p_s$ and $q_{i+1} = p_t$. In other words, subpath $P(s,t)$ of P has been simplified by segment $q_i q_{i+1}$ in Q. Then, Q is an approximation of P and can be stored using smaller memory size, at expense of losing the accuracy of P.

Assume that *err* is the criterion or error function used to compare similarity of Q and P or to evaluate quality of this simplification. Using this error function, the error of a segment $q_i q_{i+1}$ is denoted by $err(q_i q_{i+1})$ and is defined to be the error of approximating subpath $P(s,t)$ by the segment $q_i q_{i+1}$ under the error function *err*. The error of the simplification Q is then defined to be either the maximum or the sum of the error of its links $q_i q_{i+1} (0 \leq i \leq l)$ and are respectively denoted by $err^{Max}(Q)$ and $err^{Sum}(Q)$. So, having the error function *err*, we try to minimize $err^{Max}(Q)$ or $err^{Sum}(Q)$ which are called max-simplification and sum-simplification, respectively. The max-simplification is usually used for distance-distortion error functions while the sum-simplification is used for area-distortion metrics.

Definition of an error function *err* usually depends on the underlying application. The Hausdorff distance, err_h, as the main distance-distortion error function is the metric used in many simplification algorithms. For a segment $q_i q_{i+1}$ which is the simplification of the subpath $P(s,t)$, $err_h(q_i q_{i+1})$ is defined to be the maximum Euclidean distance of points $p_s, p_{s+1}, \ldots, p_t$ from the segment $q_i q_{i+1}$. The Euclidean distance of a point p from a segment qr, denoted by $d(p,qr)$, is defined to be $|pt|$ where t is the point on qr that minimizes $|pt|$. For example, $err_h(p_r p_s)$ in Fig. 1 is equal to $|p_k p_r|$ which is the shortest distance between p_k and points of segment $p_r p_s$, and $err_h(p_s p_t)$ of this figure is equal to $d_o(p_m, p_s p_t)$, where $d_o(p_m, p_s p_t)$ is defined as the orthogonal distance of p_m from the supporting line of $p_s p_t$ (the line that passing over a segment is called its supporting line).

Sum-area, err_a, is the main area-distortion error function. Here, $err_a(q_i q_{i+1})$ is defined to be the area of the region contained between the previously defined $q_i q_{i+1}$ and $P(s,t)$.

The error functions used so far do not take the position of observers into account. We consider observer-dependent simplification of visibility polygons, where the vertices of the path that are closer to the observer are more important to keep than the farther points. Precisely, assume that $P = p_0, p_1, p_2, \ldots, p_n, p_0$ is $VP(q)$ for a point observer q. Then, we need an approximating error function that considers the distance between the points of P and the observer q. Moreover, we need simplification algorithms based on this error function to compute an l-simplification of P. In this paper, we assume that $VP(q)$ is given as the problem input. Also, we assume that the initial (and final) point p_0 is given as problem input ($VP(q)$ is a closed simple polygon and anyone of its vertices can be taken as p_0. However, different simplifications are obtained for different choices of p_0 and we will

need more processing time to find the best simplification among them. This will increase the running time of our algorithms. Therefore, we assume that p_0 is given as a problem input). We only consider the *min-k* simplification in this paper from which the *min-δ* simplification is obtained by a simple binary search on δ valuses.

Our Results We define observer-dependent approximating error functions for $VP(q)$ that depend on the position of q for both distance-distortion and area-distortion criteria. For these simplification error functions, we first provide an $O(n^2)$ algorithm for max-simplification of $VP(q)$ using the proposed observer-dependent distance-distortion error function and an $O(n^3)$ algorithm for sum-simplification of $VP(q)$ using the proposed observer-dependent area-distortion error function. In these methods, we compute the associated error of the $O(n^2)$ possible links in respectively $O(n^2)$ and $O(n^3)$ time. Having these links, the former simplification is done using the general algorithm of Imai and Iri [17] and the latter is done using the algorithm presented by Bose et al. [5].

Then, we improve the max-simplification of $VP(q)$ under the proposed observer-dependent distance-distortion error function to $O(n^{4/3+\delta})$. In this improvement we employ the method used by Agarwal and Varadarajan [3] in which the $O(n^2)$ links are maintained implicitly in a clique cover. Also, we define the unrestricted version of this problem and propose an efficient method for solving it in $O(n \log k)$ time where k is the complexity of the resulting simplification.

We further consider the cases in which the observer behaves like a radar inside a planar environment. Here, the observer sweeps its neighborhood circularly, 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 these points. Therefore, it is necessary to approximate the exact visibility polygon by another polygon of smaller number of vertices.

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

To the best of our knowledge, the result of this paper is the first in this area and the efficiency of the proposed methods is as well as the best current simplification algorithms—that do not take into account the observer position. The only similar work to the results of this paper is the rendering based simplification of Buzer [7], which depends on the resolution of the display screen without considering the observer position. There are several interesting open directions in applying and extending this notion.

The rest of this paper is organized as follows: in Sect. 2, our *observer-dependent* simplification error function is described. In Sect. 3, we propose simplification algorithms for point observer visibility polygon in an offline setting and in Sect. 4, we solve the problem in streaming model.

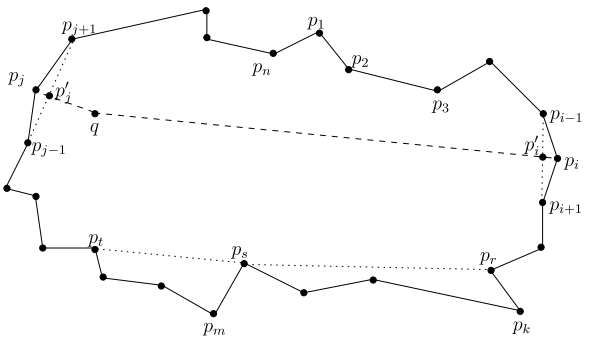


Fig. 1 Simplifying visibility polygon

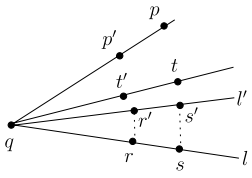
2 Observer-Dependent Simplification Error Functions

The Hausdorff error function only depends on the initial and the simplified paths and therefore, is not proper for observer-dependent simplification of visibility polygons in which the position of the observer has an important effect. We illustrate this claim by an example. Assume that $P = p_1, p_2, \ldots, p_n, p_1$ of Fig. 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 very far away from q. Also, assume that we are asked to simplify P by removing one vertex and p_i and p_j are the only choices for elimination. If we remove p_j, the actual position of p_j and its position in the resulting simplification (from the viewpoint of q) differs by $|p_j p'_j|$. However, this deviation occurs at a distance of $|p_j q|$ from the observer. In contrast, if we eliminate vertex p_i, our deviation from the initial polygon is $|p_i p'_i|$ which happens at distance $|p_i q|$ from the observer. Since, p_i was assumed to be too much farther from q than p_j, the latter simplification is better (from the viewpoint of q) even if $|p_i p'_i|$ was *slightly* larger than $|p_j p'_j|$. However, if we use Hausdorff error function, p_j will be removed if $|p_i p'_i|$ is a bit larger than $|p_j p'_j|$. The same argument is true for all distance and area distortion simplification error functions.

We therefore need to formally define a new error function that includes the position of the observer in computing the error of a simplification. Moreover, in order to use current simplification algorithms, the new function must be computed easily and can be simply plugged into these algorithms.

To define a proper simplification criterion, we need to consider how distance affects visibility or illumination. Assume that r and s are two points on $VP(q)$. Theoretically, while r and s are distinct points they are seen as different points on $VP(q)$. But, a realistic observer has a limited *vision distinction power*, *i.e.* for small values of $|rs|$ the observer sees them as a single point. This means that when the angle between segments qr and qs is too small then the observer can not distinguish them. This threshold angle is usually independent of the distance between the visible points and the observer. We call this threshold angle the vision distinction power of the observer. Therefore, a good candidate criterion for comparing visibility brightness of two segments is the angles between the segments that connect their endpoints to the observer. But, we always need to compare this for segments that lie on the rays em-

Fig. 2 Vision distinction power



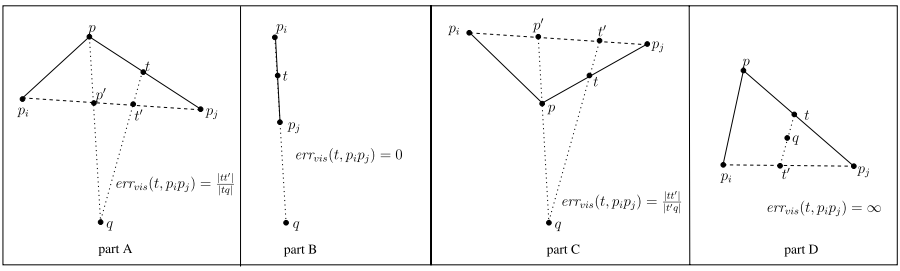


Fig. 3 Observer-dependent simplification error

anating from the observer for which these angles are zero. For example we need to compare this for the segments $|p_j p'_j|$ and $|p_i p'_i|$ in the above scenario.

Let's look at the angle criterion again. For parallel segments rr' and ss' shown in Fig. 2, these angles are equal. For these segments we have $\frac{|rr'|}{|ss'|} = \frac{|qr|}{|qs|}$, and therefore, $\frac{|rr'|}{|qr|} = \frac{|ss'|}{|qs|}$. These relations are still valid if l and l' are too close or even overlap. Therefore, for any two segments pp' and tt' on two rays emanating from q (see Fig. 2) a proper visibility brightness comparison is to compare $\frac{|pp'|}{|pq|}$ and $\frac{|tt'|}{|tq|}$.

Now, we are ready to define our observer-dependent simplification error functions. Assume that we are to approximate the path $p_i pp_j$ (see Fig. 3.A), a part of $VP(q)$, by the segment $p_i p_j$. From the viewpoint of q, p is mapped to p' and other points of segments $p_i p$ and pp_j are mapped to their corresponding points of segments $p_i p'$ and $p' p_j$. In such simplifications, the corresponding observer-dependent error of a point t on the path $p_i pp_j$ is denoted by $err_{vis(q)}(t, p_i p_j)$ (or simply by $err_{vis}(t, p_i p_j)$ whenever q is known from context) and is defined to be $\frac{|tt'|}{|tq|}$ where t' is the intersection point of segments tq and $p_i p_j$. This means that at distance $|tq|$, we have deviated from the initial path by a value of $|tt'|$. This definition is also used for paths of more internal vertices.

Three situations are not covered by this definition and are handled as follows. If p_i, p_j and q are collinear (see Fig. 3.B), p and all other points of the polygon boundary from p_i to p_j must also lie on the segment $p_i p_j$. The reason is that the path $P(i, j)$ is a part of $VP(q)$ which is a star-shaped simple polygon and q is a core point of $VP(q)$. Then, segments tq and $p_i p_j$ overlap and do not have a single intersection point. In this situation, there is no difference between subpath $P(i, j)$ and its corresponding subpath in the simplification. Therefore, the corresponding error of the points in subpath $P(i, j)$ is defined to be zero.

Another problem of the definition of $err_{vis}(t, p_i p_j)$ is that the segments $p_i p_j$ and tq do not necessarily intersect, *i.e.*, point t' does not always exist. Since, $VP(q)$ is a

star-shaped simple polygon and t lies on the boundary of this polygon between p_i and p_j, the supporting line of pq always intersects $p_i p_j$. This intersection point may lie outside the segment tq (see Fig. 3.C). Here, from the viewpoint of q, t is mapped to t' which is this intersection point. In these cases, the observer-dependent error of point t is defined to be $\frac{|tt'|}{|t'q|}$. Unlike the cases shown in Fig. 3.A, we do not use $\frac{|tt'|}{|tq|}$ as the error value here. The reason is that the value of $\frac{|tt'|}{|tq|}$ in the former cases (Fig. 3.A) is always equal to something between zero and one. This is also true for $\frac{|tt'|}{|t'q|}$ in the latter cases (Fig. 3.C) as well, while, the value of $\frac{|tt'|}{|tq|}$ ranges from zero to ∞ here.

Finally, we are supposed to simplify $VP(q)$ for which q is an internal point and we expect to have this property in the simplified version of $VP(q)$. To support this requirement, we must prevent segments that contradict this condition (like the segment $p_i p_j$ in Fig. 3.D) to appear in the simplification. In such cases, there are some points like t on subpath $P(i, j)$ in Fig. 3.D, such that the observer q lies between t and t'. We define the observer-dependent error of such points to be infinite. Then, the error of segment $p_i p_j$ will also be infinite (to be defined later) and this infinite error value prevents such segments to be included in the resulting simplification.

According to the above definition of err_{vis} function, it is clear that $err_{vis}(p, p_i p_j)$ is either infinite or a number between zero and one. Using this definition, we propose our observer-dependent error functions for both distance- and area-distortion cases.

2.1 Observer-Dependent Distance-Distortion Error

The observer-dependent distance-distortion error of a segment $p_i p_j$ which is a simplification of a subpath $P(i, j)$ of $VP(q)$ is defined to be $\max(err_{vis(q)}(t, p_i p_j))$ over all points t on the subpath $P(i, j)$. This error is denoted by $err_{dvis(q)}(p_i p_j)$ (or simply by $err_{dvis}(p_i p_j)$ if q is known from context).

There is a close relation between this observer-dependent error function and the *width* notion. The width of a set of points with respect to a given direction $\overrightarrow{d}$ is the minimum distance between a pair of lines being parallel to $\overrightarrow{d}$ that encloses the point set. Let $P_L(i, j)$ (resp. $P_U(i, j)$) be the set of points of the subpath $P(i, j)$ that lie in the closed half plane defined by the supporting line of $p_i p_j$ which contains (resp. does not contain) the observer q. We denote by $w_L(i, j)$ (resp. $w_U(i, j)$) the width of the points of $P_L(i, j)$ (resp. $P_U(i, j)$) with respect to the direction of $\overrightarrow{p_i p_j}$. We have,

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

$$err_{vis}(p_i p_j) = \max\left(\frac{w_U(i, j)}{d_o(q, p_i p_j) + w_U(i, j)}, \frac{w_L(i, j)}{d_o(q, p_i p_j)}\right).$$

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

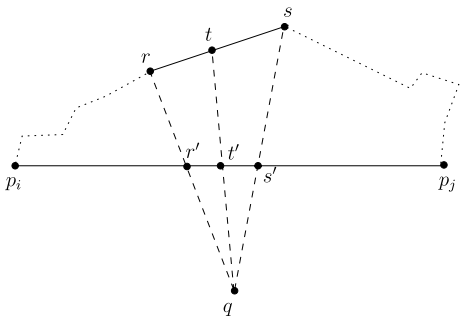


Fig. 4 Observer-dependent area-distortion error

A direct consequence of this lemma is that the associated error of a segment $p_i p_j$ belongs to a vertex p_k with $(i \leq k \leq j)$. Using this result, we can simply compute the corresponding error of any segment $p_i p_j$ only by checking vertices of the subpath $P(i, j)$.

2.2 Observer-Dependent Area-Distortion Error

According to the relation between distance and area, the observer-dependent area-distortion error of the segment $p_i p_j$ is defined to be $\sum(err_{vis(q)}(t, p_i p_j))$ over all points t on the subpath $P(i, j)$. This error is denoted by $err_{avis(q)}(p_i p_j)$ (or simply by $err_{avis}(p_i p_j)$ if q is known from context).

To compute the value of $err_{avis}(p_i p_j)$ we must compute sum of infinite values. This is done by solving an integral. Assume that the segment rs, shown in Fig. 4, is a part of $P(i, j)$ simplified by $p_i p_j$. The sum of the observer-dependent error of all points of rs is denoted by $err_{avis}(rs, p_i p_j)$. Without loss of generality, we assume that $p_i p_j$ lies on x axis. Then, $err_{avis}(rs, p_i p_j) = \int_{x_{r'}}^{x_{s'}} \frac{|tt'|}{|tq|} dx$ where $x_{r'}$ and $x_{s'}$ are respectively x coordinates of r' and s'. We can solve this integral as follows.

Assume that $ax + by + c = 0$ is the equation of the supporting line of rs and $x_q = m$ and $y_q = n$. The point t lies on both segments rs and tq and, according to the above assumption, $x_{t'} = x$ and $y_{t'} = 0$. Then, we can compute x_t and y_t as follows:

$$\begin{cases} y_t - n = \dfrac{n-0}{m-x}(x_t - m), \quad ax + by_t + c = 0 \\ \Rightarrow \quad \begin{cases} y_t = \dfrac{nc + anx}{ax - am - bn}, \quad x_t = \dfrac{cm - cx - bnx}{ax - am - bn}. \end{cases} \end{cases}$$

Therefore,

$$err_{avis}(rs, p_i p_j) = \int_{x_{r'}}^{x_{s'}} \frac{|tt'|}{|tq|} dx = \int_{x_{r'}}^{x_{s'}} \frac{\sqrt{(x - x_t)^2 + (0 - y_t)^2}}{\sqrt{(m - x_t)^2 + (n - y_t)^2}} dx.$$

The closed form of this integral is,

$$\mathcal{F}(x) = \int \frac{|tt'|}{|tq|} dx = \frac{(t - x_t)\sqrt{(t - x_t)^2 + y_t^2} + y_t^2 \ln(t - x_t + \sqrt{(t - x_t)^2 + y_t^2})}{2\sqrt{(m - x_t)^2 + (n - y_t)^2}}$$

from which the value of $err_{avis}(rs, p_i p_j)$ is computed: $err_{avis}(rs, p_i p_j) = \mathcal{F}(x_{s'}) - \mathcal{F}(x_{r'})$.

The same computation can be done for the other case of observer-dependent error shown in Fig. 3.C. Having this formula, we can compute $err_{avis}(p_i p_j)$ by summing the values of $err_{avis}(rs, p_i p_j)$ for all segments rs of $P(i, j)$.

3 Simplification Algorithms

In Sect. 2, we defined two observer-dependent error functions and described how to compute $err_{dvis}(p_i p_j)$ and $err_{avis}(p_i p_j)$ for candidate links in the simplification. Naively, we can compute err_{dvis} and err_{avis} of each link in $O(n)$ time. There are $O(n^2)$ candidate links. Then, the total time complexity of computing the error of all links is $O(n^3)$.

Having the error of all links, there are efficient general algorithms to solve *min-k* version of both max-simplification and sum-simplification problems. As a general approach, Imai and Iri [17] modeled the *min-k* version of the max-simplification problem by a directed acyclic graph G over the vertices of the path $P = p_0, p_1, \ldots, p_n$. For any $p_i \in P$, there is a vertex p_i in G and an edge $\overrightarrow{p_i p_j}$ is added to this graph if and only if (iff) $err(p_i p_j)$ is not greater than the allowed error threshold. The shortest path from p_0 to p_n in G is then the solution of this version of simplifying P. Also, Bose et al. [5] proposed a dynamic programming approach to solve the max-simplification problem. We can use these algorithms along with the proposed observer-dependent error functions to solve simplification problems. Efficient implementation of the DAG approach can be run in $O(n^2)$ [8], and running time of the dynamic programming approach is $O(kn^2 + n^{2+\delta})$ [5]. Therefore, the naive method for computing error of the links, $O(n^3)$, is our bottleneck in obtaining efficient simplification algorithms under the observer-dependent error functions.

In this paper, we do not do better for err_{avis}, but for err_{dvis} we propose $O(n^2)$ and $O(n^{4/3+\delta})$ algorithms to solve the restricted *min-k* version of the max-simplification problem and an $O(n \log n)$ algorithm for its unrestricted version. In the rest of this paper we only consider the observer-dependent distance-distortion error function and solve the *min-k* version of the max-simplification problem.

3.1 The $O(n^2)$ Algorithm

A segment $p_i p_j$ can appear in *min-k* simplification of a path P if for all points p_m $(i < m < j)$ we have $err_{dvis}(p_m, p_i p_j) \leq e$ where e is the given error threshold. This means that for any point p_m $(i < m < j)$, $p_i p_j$ must intersect the segment $p'_m p''_m$ (see Fig. 5) where p'_m is the point on segment $p_m q$, such that $|p'_m p_m| = e|p_m q|$ and p''_m is the point on the supporting line of $p_m q$ and in the opposite side of p_m compared to p'_m such that $|p''_m p_m| = e|p''_m q|$. Equivalently, edge $\overrightarrow{p_i p_j}$ is added to the DAG G of the general algorithm of Imai and Iri [17] iff $p_i p_j$ intersects segments $p'_m p''_m$ for all vertices p_m with $i < m < j$. The segment $p'_m p''_m$ is called *the tolerance zone* of p_m with respect to the observer q and error threshold e.

Therefore, to decide on $p_i p_j$ (adding edge $\overrightarrow{p_i p_j}$ to G or not) we should consider the tolerance zone of all vertices of P that are between p_i and p_j. Then, to

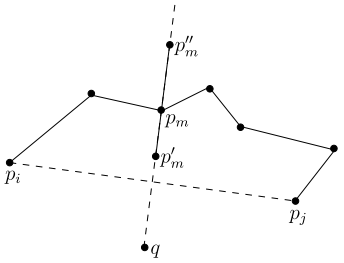


Fig. 5 Observer-dependent tolerance zone

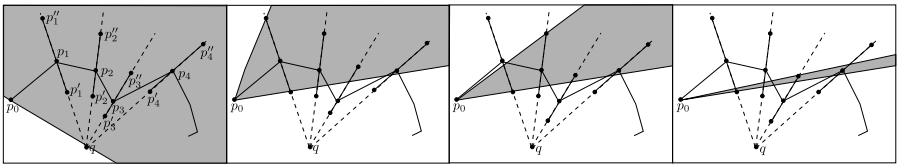


Fig. 6 The gray region is the permitted region of p_0 as we proceed considering the points p_1 to p_4

```
1:  procedure BuildDAG(P : path of n vertices, e : error threshold)
2:  for i = 1 to n − 1
3:      PR_{p_i} = hp(q, p_i, p_{i+1})
4:      for j = i + 1 to n do
5:          if p_j ∈ PR_{p_i} then
6:              add p_i p_j to G
7:          end if
8:          PR_{p_i} = PR_{p_i} ⋂ ◁ p'_j p_i p''_j
9:      end for
10: end for
11: return G
```

Fig. 7 Building DAG for the general simplification algorithm

decide on $p_i p_{j+1}$ we can use the information prepared when deciding on $p_i p_j$. We do this by defining a *permitted region* for p_i, denoted by PR_{p_i}, which is a part of the plane with this property that, when deciding on $p_i p_j$, this edge is added to G iff p_j lies inside PR_{p_i}. Trivially, the initial value of PR_{p_i} is the half plane defined by qp_i containing p_{i+1}—this half-plane is denoted by $hp(q, p_i, p_{i+1})$. The reason is that $err_{dvis}(p_i p_{i+1}) = 0$ and regardless of the position of p_{i+1}, $\overrightarrow{p_i p_{i+1}}$ is always added to G. Let $\lhd pqr$ denote the wedge defined by the edges $\overrightarrow{qp}$ and $\overrightarrow{qr}$ that contains segment pr. Then, $\overrightarrow{p_i p_{i+2}}$ must be added to G iff p_{i+2} lies inside $\lhd p'_{i+1} p_i p''_{i+1}$. Therefore, after considering p_{i+1}, PR_{p_i} must be updated to $\lhd p'_{i+1} p_i p''_{i+1}$. Then, when we consider p_{i+2}, PR_{p_i} has the property that $\overrightarrow{p_i p_{i+2}}$ is added to G iff p_{j+2} lies inside PR_{p_i}. In Fig. 6 we have shown how the permitted region of the point p_0 changes when we proceed to decide on the segments $p_0 p_k$ $(0 < k < 5)$. According to this figure, edges $\overrightarrow{p_0 p_1}$, $\overrightarrow{p_0, p_2}$ and $\overrightarrow{p_0 p_4}$ are added to G, but, $\overrightarrow{p_0 p_3}$ is not added to G. Using this approach, we can build G in total $O(n^2)$ time. The pseudo code shown in Fig. 7 depicts details of this method.

Using this method, our observer-dependent distance-distortion error function can be used along with current efficient line simplification algorithms without increasing their running time complexity.

Theorem 1 *The visibility polygon of a point observer inside a planar domain of total n vertices can be simplified according to the observer-dependent distance-distortion error function in* $O(n^2)$ *time.*

3.2 The $O(n^{4/3+\delta})$ Algorithm

Agarwal and Varadarajan [3] have proposed the only sub-quadratic simplification algorithm for L_1 and uniform error functions. Their algorithm is based on preparing a clique cover for the DAG G of Imai and Iri approach [17].

A set $\mathcal{G} = \{G_1(V_1, E_1), \ldots, G_l(V_l, E_l)\}$ is a clique cover of a directed graph $G = (V, E)$ if,

- Each G_i is a subgraph of G,
- Each G_i is a complete bipartite graph and if V_{i_1} and V_{i_2} are the vertex classes of G_i then each edge of E_i is directed from a vertex in V_{i_1} to a vertex in V_{i_2},
- $E = \bigcup_{i=1}^{l} E_i$,
- $E_i \cap E_j = \emptyset$ for all $1 \le i \le l$, $1 \le j \le l$, $i \ne j$.

Then, we need $O(|\mathcal{G}|) = O(\sum_{i=1}^{l} |V_i|)$ space to store $\mathcal{G}$. Having this compressed version of G, we can find the shortest path between any two vertices of G in time $O(|V| + |\mathcal{G}|)$ [3].

They proposed a method for finding such a clique cover of the simplification DAG for solving max-simplification under L_1 and uniform metric. In this section, we apply this method for solving *min-k* simplification under observer-dependent distance-distortion error function.

Assume that $VP(q) = C = \langle p_1, p_2, \ldots, p_n \rangle$ is the given visibility polygon and G is the corresponding DAG. We compute a clique cover $\mathcal{G}$ of G by a divide and conquer scheme. Assume that $C_1 = \langle p_1, p_2, \ldots, p_{\lfloor \frac{n}{2} \rfloor} \rangle$ and $C_2 = \langle p_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, p_n \rangle$ are the two parts of C in divide step and $\mathcal{G}_1$ and $\mathcal{G}_2$ are respectively their clique covers. Then, we need to prepare a clique cover $\mathcal{G}_{12}$ for edges $E_{12} = \{\overrightarrow{p_i p_j} | \overrightarrow{p_i p_j} \in G, p_i \in C_1, p_j \in C_2\}$ in the merge step. If we do this, $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_{12}$ will be a clique cover for G. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ are computed recursively, it is enough to describe a method for building $\mathcal{G}_{12}$.

In Sect. 3.1, we defined PR_{p_i} to be initially the half-plane $hp(q, p_i, p_{i+1})$ and its value is updated as vertices $p_{i+1}, p_{i+2}, \ldots$ are processed. Let $C_1 = \langle p_1, p_2, \ldots, p_{\lfloor \frac{n}{2} \rfloor} \rangle$, $\overline{C_2} = \langle p_n, p_{n-1}, \ldots, p_{\lfloor \frac{n}{2} \rfloor + 1} \rangle$, $p_i \in C_1$ and $p_j \in \overline{C_2}$. We define $cone(p_i)$ to be equal to PR_{p_i} after processing $p_{\lfloor \frac{n}{2} \rfloor}$ and $cone(p_j)$ to be equal to PR_{p_j} after processing $p_{\lfloor \frac{n}{2} \rfloor + 1}$. The following lemma defines the relation between edges of E_{12} and *cone* of the vertices.

Lemma 2 $\overrightarrow{p_i p_j} \in E_{12}$ *iff* $p_j \in cone(p_i)$ *and* $p_i \in cone(p_j)$.

Proof In Sect. 3.1, we showed that $\overrightarrow{p_i p_j} \in G$ iff $p_i p_j$ intersects all segments $p'_m p''_m$ for $i < m < j$. Trivially, if $p_i p_j$ intersects these segments then $p_j \in cone(p_i)$ and $p_i \in cone(p_j)$. On the other hand, if $p_j \in cone(p_i)$ and $p_i \in cone(p_j)$, $p_i p_j$ will intersect all segments $p'_m p''_m$ for $i < m < j$ which results that $p_i p_j$ is an edge in G and therefore, it is an edge in E_{12}. □

Assume that $p_i l$ and $p_i l'$ are the half lines defining $cone(p_i)$ and α is the convex angle defined by these half lines. We denote by $wedge(p_i)$ the wedge of angle α defined by the supporting lines of $p_i l$ and $p_i l'$. Therefore, $wedge(p_i)$ is the union of $cone(p_i)$ and its inverse relative to p_i. The following lemma defines the relation between *cone* and *wedge* of the vertices.

Lemma 3 *The following statements are equivalent*:

1 $\overrightarrow{p_i p_j} \in E_{12}$.
2 $p_j \in cone(p_i)$ *and* $p_i \in cone(p_j)$.
3 $p_j \in wedge(p_i)$, $p_i \in wedge(p_j)$, $p_j \in hp(q, p_i, p_{i+1})$ *and* $p_i \in hp(q, p_j, p_{j-1})$.

Proof The proof of $1 \Leftrightarrow 2$ was given in Lemma 2. Proof of $2 \Rightarrow 3$ is directly derived from the definition of *cone* and *wedge*. Also, from these definitions we have $cone(p_i) = hp(q, p_i, p_{i+1}) \cap wedge(p_i)$ and $cone(p_j) = hp(q, p_j, p_{j-1}) \cap wedge(p_j)$ from which the relation $3 \Rightarrow 2$ results. □

According to Lemma 3, the set of edges in E_{12} can be declared as follows:

$$E'_{12} = \{\overrightarrow{p_i p_j} | p_i \in C_1, p_j \in \overline{C_2}, p_j \in hp(q, p_i, p_{i+1})\},$$
$$E''_{12} = \{\overrightarrow{p_i p_j} | \overrightarrow{p_i p_j} \in E'_{12}, p_i \in hp(q, p_j, p_{j-1})\},$$
$$E_{12} = \{\overrightarrow{p_i p_j} | \overrightarrow{p_i p_j} \in E''_{12}, p_i \in wedge(p_j), p_j \in wedge(p_i)\}.$$

In the remainder of this section, we first describe a method to compute $cone(p)$ for all $p \in C_1$ and $p \in \overline{C_2}$. Then, we propose a method for building $\mathcal{G}_{12}$. The latter is done by first computing a clique cover $\mathcal{G}'_{12}$ of edges in E'_{12}. Then, for each clique item $(A, B) \in \mathcal{G}'_{12}$ we produce a clique cover of vertices $(a, b) \in A \times B$ where $(a, b) \in E''_{12}$. This results a clique cover $\mathcal{G}''_{12}$ for edges of E''_{12}. Finally, for each clique $(A', B') \in \mathcal{G}''_{12}$, we build a clique cover of edges $(a, b) \in A' \times B'$ where $a \in wedge(b)$ and $b \in wedge(a)$. Trivially, this is a clique cover $\mathcal{G}_{12}$ of edges E_{12}.

Lemma 4 *We can compute* $cone(p_i)$ *of all vertices* $p_i \in C_1$ *and* $p_i \in \overline{C_2}$ *in* $O(n \log n)$ *time*.

Proof Assume that $p_i \in C_1$ and we have already computed $cone(p_k)$ for $i \leq k \leq \lfloor \frac{n}{2} \rfloor$ by induction. We define L_i to be the lower envelope of the convex hull of the points $\{p''_i, p''_{i+1}, \ldots, p''_{\lfloor \frac{n}{2} \rfloor}\}$ compare to the supporting line of $p_i p_{\lfloor \frac{n}{2} \rfloor}$ and U_i to be the upper envelope of the convex hull of the points $\{p'_i, p'_{i+1}, \ldots, p'_{\lfloor \frac{n}{2} \rfloor}\}$. Assume that we have computed L_i and U_i by induction.

For the new vertex p_{i-1}, we compute the tangent half lines from this point to L_i and U_i. If the tangent to L_i lies below the tangent to U_i, $cone(p_{i-1})$ is empty. Otherwise, $cone(p_{i-1})$ is the region contained between these tangents. Trivially, any ray in $cone(p_{i-1})$ intersects $p''_k p'_k$ tolerance zones for $i \leq k \leq \lfloor \frac{n}{2} \rfloor$ and any ray outside $cone(p_{i-1})$ does not have this property. Therefore, $cone(p_{i-1})$ have been computed correctly. We must also compute L_{i-1} and U_{i-1} to be used for point p_{i-2}. This is done by using current incremental or dynamic convex hull algorithms [20, 22]. L_{i-1} and U_{i-1} is obtained by applying required changes to L_i and U_i when p''_{i-1} and p'_{i-1} are added to their corresponding convex hulls. The same method can be used for points $p_i \in \overline{C_2}$.

Finding tangent lines and convex hull updates can be done in $O(\log n)$ [20, 22]. Therefore, the total time complexity of computing $cone(p_i)$ for all vertices $p_i \in C_1$ and $p_i \in \overline{C_2}$ is $O(n \log n)$. □

We can build the clique covers for E'_{12} and E''_{12} according to the following lemmas.

Lemma 5 *We can compute a clique cover of size $O(n \log n)$ for E'_{12} in $O(n \log n)$ time.*

Proof We put vertices $p_j \in \overline{C_1}$ into a range searching data structure so that the following query can be answered efficiently: For a vertex $p_i \in C_1$ find all vertices $p_j \in \overline{C_1}$ that lie in half-plane $hp(q, p_i, p_{i+1})$.

This is done by building a binary search data structure over the vertices of $\overline{C_1}$. Since vertices of $p_j \in \overline{C_1}$ are radially sorted around q and the half-plane queries pass through this point the size of this search data structure is $O(|\overline{C_1}|) = O(n)$. Then, the answer of a query for vertex p_i is given as union of $O(\log n)$ canonical and disjoint subset of $\overline{C_1}$ in $O(\log n)$ time. The total size of all canonical subsets of this data structure is $O(n)$ and can be constructed in $O(n \log n)$.

Having this data structure, for each vertex $p_i \in C_1$ we run the query and for each canonical subset B_k of this data structure, a set A_k is built that contains those vertices of C_1 that B_k exists in their query result. If $A_k \neq \emptyset$, (A_k, B_k) is selected as a member of the clique cover $\mathcal{G}'_{12}$. It is simple to verify that $\mathcal{G}'_{12}$ is a correct clique cover for E'_{12}.

The total size of the canonical subsets B_i is $O(n)$ and each vertex of C_1 must be added to at most $O(\log n)$ items of A_k's. Therefore, the total size of this clique cover is $O(n \log n)$ and can be constructed in $O(n \log n)$ time. □

Lemma 6 *We can compute a clique cover of size $O(n \log^2 n)$ for E''_{12} in $O(n \log^2 n)$ time.*

Proof Based on the relation between E'_{12} and E''_{12}, for each clique (A_k, B_k) of the clique cover of the edges of E'_{12}, as prepared according to Lemma 5, we build a clique cover for the set of edges in $(A_k \times B_k) \cap E''_{12}$.

Consider a clique $(A_k, B_k) \in \mathcal{G}'_{12}$. We build the range searching data structure described in Lemma 5 over the vertices of A_k so that the following query can be answered efficiently: For a vertex $p_j \in B_k$ find all vertices $p_i \in A_k$ that lie in the

half-plane $hp(q, p_j, p_{j-1})$. As describe in the proof of Lemma 5, we will build a clique cover for the set of edges $(A_k \times B_k) \cap E''_{12}$ in $O(|A_k \cup B_k| \log(|A_k \cup B_k|))$ time whose size is also $O(|A_k \cup B_k| \log(|A_k \cup B_k|))$.

If we do this for all cliques $(A_k, B_k) \in \mathcal{G}'_{12}$, the total size of the resulting clique covers will be $O(n \log^2 n)$ and these cliques are built in total $O(n \log^2 n)$ time. Trivially, each edge of this clique cover exists in E''_{12} and vice versa. □

Agarwal and Varadarajan [3] proposed a method for building a clique cover for $E = \{(a, b) | a \in A, b \in B, a \in wedge(b), b \in wedge(a)\}$ which is based on mapping lines in $wedge(p)$ to dual space and computing segment intersection in this dual space. The size of this clique cover is $O(n^{4/3+\delta})$ and it can be computed in $O(n^{4/3+\delta})$ time where n is the number of vertices in A and B.

Using this method and having the clique cover of edges of E''_{12} as described in Lemma 6, we can build the clique cover $\mathcal{G}_{12}$ for edges of E_{12} as follows. For each $(A'_k, B'_k) \in \mathcal{G}''_{12}$ of the clique cover of E''_{12}, we build a clique cover for the edges of $(A'_k \times B'_k) \cap E_{12}$ using the method proposed in [3]. It is simple to prove that the total size of this cover and its building running time for all cliques of $\mathcal{G}''_{12}$ is $O(n^{4/3+\delta})$.

Combining everything together, we have the following relation for the total running time and required space for our divide and conquer algorithm to build a clique cover for edges in G.

$$\begin{aligned} S(n) &\leq 2S(n/2) + c \cdot n^{4/3+\delta} \\ &= O(n^{4/3+\delta}) \end{aligned}$$

Lemma 7 *There is a clique cover of size $O(n^{4/3+\delta})$ for the simplification graph of a visibility polygon of size n under the observer-dependent distance-distortion error function. This clique cover can be constructed in $O(n^{4/3+\delta})$.*

Using the result of Lemma 7 and doing a shortest path in this clique cover, we can simplify $VP(q)$ in subquadratic time.

Theorem 2 *The visibility polygon of a point observer can be simplified using the observer-dependent distance-distortion error function in time $O(n^{4/3+\delta})$.*

3.3 Unrestricted Simplification

In previous simplifications, we need the vertices of the simplification to be selected among vertices of the initial visibility polygon. In this section, we solve the unrestricted version of the line simplification problem to simplify $VP(q)$. In this problem, we are asked to simplify $VP(q)$ by a polygon Q of minimum number of vertices such that the error of this simplification (the maximum observer-dependent distance-distortion error of the vertices of $VP(q)$) supports a give error threshold e. Here, the vertices of Q are not restricted and can be selected anywhere in the plane.

To support the error thereshold e, Q must intersect the tolerance zones $p'_i p''_i$ of all vetices $p_i \in VP(q)$ where $p'_i p''_i$ is computed with respect to q and e. As shown in Fig. 8, this problem is reduced to the problem of finding minimum-link

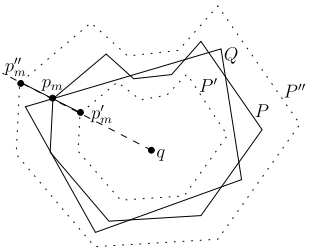


Fig. 8 Unrestricted visibility simplification is equivalent to finding minimum-link loop

loop $Q = q_0, q_1, \ldots, q_k$ that separetes the simple polygons $P' = p'_0, p'_1, \ldots, p'_n$ and $P'' = p''_0, p''_1, \ldots, p''_n$. This minimum-link problem is a well-known problem for which a $O(n \log k)$ time algorithm exists [15]. Therefore,

Theorem 3 *The unrestricted version of simplifying the visibility polygon of a point observer using the observer-dependent distance-distortion error function can be solve in time $O(n \log k)$ where k is the number of the vertices of the resulting simplification.*

4 Observer-Dependent Simplification in Streaming Model

In the previous section, we considered the cases where we have all the path vertices in memory and we can do more than a single processing pass over them. In this section we consider this simplification in streaming model in which we have a single pass over the path vertices. 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 solves only the *min-δ* version of the line simplification problem. Because of the large number of 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 it.

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

- *err* must be a *c-monotone* error function on $P(n)$ for any $n > 0$. This means that for any two segments $p_i p_j$ and $p_l p_m$ such that $i \leq l \leq m \leq j$ and p_i, p_j, p_l and p_m are vertices of $P(n)$, we have

$$err(p_l p_m) \leq c \cdot err(p_i p_j).$$

  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 *err* on $P(n)$ to be defined as follows. In streaming model, we may lose some vertices of the subpath $P(i, j)$ between points p_i and p_j. Thus, we can not compute the exact value of the error function for this segment and we should approximate it. We denote the approximated error value of a segment $p_i p_j$ by $err^*(p_i p_j)$. We call the procedure that

computes this approximation our error *oracle*. An error oracle is e-approximate if for any segment $p_i p_j$ for which the oracle is called by the algorithm we have

$$err(p_i p_j) \leq err^*(p_i p_j) \leq e \cdot err(p_i p_j).$$

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. Beyond the storage needed for the simplification Q that is $O(k)$, the algorithm needs $O(k)$ storage plus the storage needed by the error oracle. The algorithm is quite simple:

Suppose we have already handled points $p_0, \ldots, p_n$. (We assume $n > k + 1$; until that moment, we can simply use all points and have zero error.) Let $Q := q_0, q_1, \ldots, q_k, q_{k+1}$ be the current simplification. The algorithm will maintain a priority queue $\mathcal{Q}$ that stores points q_i with $1 \leq i \leq k$, where the priority of a point q_i 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 $q_{k+2} := p_{n+1}$, thus obtaining a $(k+1)$-simplification of $P(n+1)$.
2. Compute $err^*(q_k q_{k+2})$ and insert q_{k+1} into $\mathcal{Q}$ with this error as priority.
3. Extract 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 $k = 2k'$ is at most ce times the error of the optimal simplification of P with k' points in non-streaming model in which we have all points in memory. So, in order to use this algorithm, we must show that our observer-dependent error function, err_{vis}, is c-monotone and we need an error oracle to approximate the error of any segment $p_i p_j$ for which the oracle is called in this algorithm.

Lemma 8 *Over the visibility polygon of a point observer, the observer-dependent error function err_{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 $err_{vis}(p_l p_m)$ 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 $VP(q)$, the supporting lines of the segments $p_l q$, $p_m q$ and $p_k q$ intersect the segment $p_i p_j$ and the supporting line of $p_k q$ intersects $p_l p_m$. There are six permutations for positions of points p_k, p'_k and p''_k on the supporting line of qp_k shown in Fig. 9. For all of these configurations we have

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

and

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

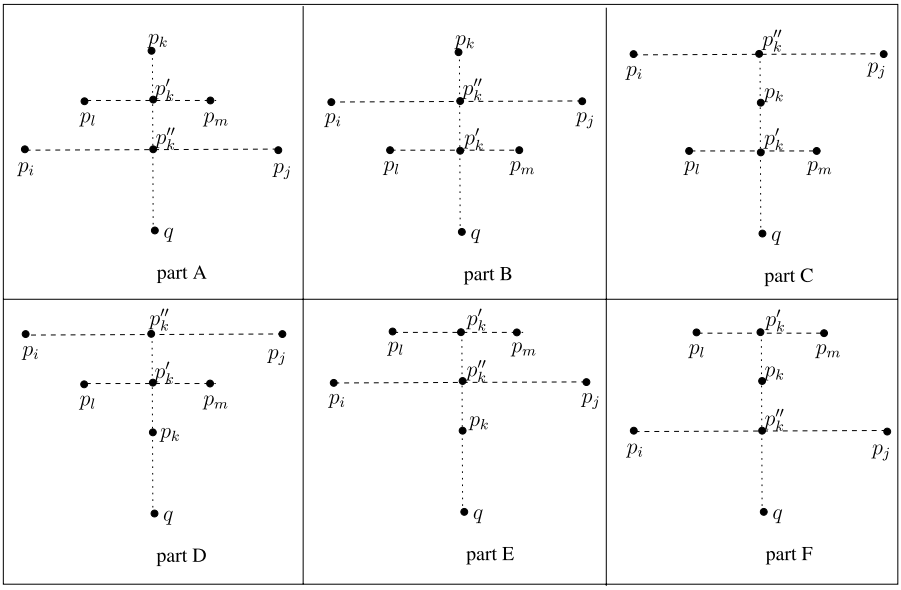


Fig. 9 The observer-dependent error function is 2-monotone

Consequently, we have $err_{vis}(p_i p_j) \geq \max(err_{vis}(p'_k, p_i p_j), err_{vis}(p_k, p_i p_j))$. We prove the lemma for all these configurations by showing that

$$\begin{aligned} err_{vis}(p_l p_m) &= err_{vis}(p_k, p_l p_m) \\ &\leq 2\max(err_{vis}(p'_k, p_i p_j), err_{vis}(p_k, p_i p_j)) \\ &\leq 2err_{vis}(p_i p_j). \end{aligned}$$

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

– Case 1 (shown in Fig. 9.A): In this configuration we have,

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

– Case 2 (shown in Fig. 9.B): Here, if $|p_k p''_k| \geq |p''_k p'_k|$ then we have,

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

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

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

So in both conditions, the lemma is valid.

– Case 3 (shown in Fig. 9.C): For this case, assume that $|p_k p''_k| = x|p_k q| = x(|p_k p'_k| + |p'_k q|)$. Then,

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

– Case 4 (shown in Fig. 9.D): See case 6.
– Case 5 (shown in Fig. 9.E): See case 6.
– Case 6 (shown in Fig. 9.F): According to the pictures, case 4 is the same as case 1, case 5 the same as case 2 and case 6 the same as case 3.

So, we proved that in all cases, $err_{vis}(p_l p_m) \leq 2err_{vis}(p_i p_j)$. We can see (in the proof of case 2) that this upper bound is tight in cases 2 and 5. □

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

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

Agarwal and Yu 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 $O(\log \frac{1}{\epsilon})$ 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 [6]. 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 describe a method for obtaining a $(1 + e)$-approximate error oracle for err_{vis} where $e = 2\epsilon + \epsilon^2$ and the value of $err_{vis}(p_i p_j)$ can be computed in $O(\log \frac{1}{\epsilon})$ time.

Assume that $\overline{w}_L(i, j)$ and $\overline{w}_U(i, j)$ are respectively the $1 + \epsilon$ approximations of $w_L(i, j)$ and $w_U(i, j)$. Then, $w_L(i, j) \leq \overline{w}_L(i, j) \leq (1 + \epsilon)w_L(i, j)$ and $w_U(i, j) \leq \overline{w}_U(i, j) \leq (1+\epsilon)w_U(i, j)$. According to Lemma 1, we need approximations for $\frac{w_U(i,j)}{d_o(q,p_i p_j)+w_U(i,j)}$ and $\frac{w_L(i,j)}{d_o(q,p_i p_j)}$ to approximate the value of $err_{vis}(p_i p_j)$.

From the $1+\epsilon$ approximation of $w_L(i,j)$ and the exact value of $d_o(q, p_i p_j)$ we have a $1+\epsilon$ approximation for $\frac{w_L(i,j)}{d_o(q,p_i p_j)}$. Combining the relation $w_U(i,j) \le \overline{w}_U(i,j) \le (1+\epsilon) w_U(i,j)$ and $d_o(q, p_i p_j) + w_U(i,j) \le d_o(q, p_i p_j) + \overline{w}_U(i,j) \le (1+\epsilon)(d_o(q, p_i p_j) + w_U(i,j))$ we have,

$$\begin{aligned}
\frac{w_U(i,j)}{(1+\epsilon)(d_o(q, p_i p_j) + w_U(i,j))} &\le \frac{\overline{w}_U(i,j)}{d_o(q, p_i p_j) + \overline{w}_U(i,j)} \\
&\le \frac{(1+\epsilon) w_U(i,j)}{d_o(q, p_i p_j) + w_U(i,j)} \\
\Rightarrow \quad \frac{w_U(i,j)}{d_o(q, p_i p_j) + w_U(i,j)} &\le (1+\epsilon) \frac{\overline{w}_U(i,j)}{d_o(q, p_i p_j) + \overline{w}_U(i,j)} \\
&\le (1+\epsilon)^2 \frac{w_U(i,j)}{d_o(q, p_i p_j) + w_U(i,j)}
\end{aligned}$$

The above relation means that $(1+\epsilon)\frac{\overline{w}_U(i,j)}{d_o(q,p_i p_j)+\overline{w}_U(i,j)}$ is a $(1+\epsilon)^2$ approximation for $\frac{w_U(i,j)}{d_o(q,p_i p_j)+w_U(i,j)}$.

Lemma 9 *There is a $(1+e)$-approximate error oracle for the observer-dependent error function on the visibility polygon of a point observer that uses $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ storage and has $O(k \log \frac{1}{\epsilon})$ amortized update time where $e = 2\epsilon + \epsilon^2$ and 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 the path $P(n) = p_0, p_1, \ldots, p_n$. Any one of the segments $q_i q_j$ where $0 \le i \le j \le k+1$, may appear in the simplification in future and we must be able to approximate $err_{vis}(q_i q_j)$. Assume that we know the approximated values of $err_{vis}(q_i q_j)$ for $0 \le i \le j \le k+1$ by induction. Also, assume that we have $k+1$ core-sets according to the algorithm of Agarwal and Yu [2] for point sets $P(q_i, p_n)$ where $0 \le i \le k$ from which we can find the extreme points in the direction of perpendicular to $q_i p_n$.

When the new point p_{n+1} is given, it is added to these core-sets and a new core-set is created for segment $q_{k+1} p_{n+1}$. To find the approximated value of $err_{vis}(q_i p_{n+1})$ for $0 \le i \le k+1$ we find the extreme points in the direction of perpendicular to $q_i p_{n+1}$ from which $(1+\epsilon)$ approximations for $w_L(i, n+1)$ and $w_U(i, n+1)$ are obtained. Having these approximations and following the above discussion, we can compute a $(1+e)$ approximation for $err_{vis}(q_i p_{n+1})$ where $0 \le i \le k+1$. Following the simplification algorithm, if q_j is the point removed from Q its associated core-set and the $O(k)$ errors of its potential links are removed as well to save space. Then, after processing the new point p_{n+1} we have the required approximations for $err_{vis}(q_i q_j)$ and $k+1$ core-sets as assumed by induction.

Overall, $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ storage is needed for the approximated values of $O(k^2)$ potential links and the $O(k+1)$ core-sets. The update of the oracle involves creation of adding a point to $O(k)$ core-sets, computing $O(k)$ extreme point from these core-sets and saving and releasing $O(k)$ approximated link errors which are done in $O(k \log \frac{1}{\epsilon})$ amortized time. □

Combining the results of Lemmas 1, 8 and 9 with 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 observer-dependent error function:

Theorem 4 *There is a streaming algorithm that maintains a $2k$-simplification for $VP(q)$ under the observer-dependent error function. This algorithm uses $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(k \log \frac{1}{\epsilon})$ amortized time and the error of the resulting simplification is not larger than $(2 + \epsilon)$ times the error of the optimal offline k-simplification.*

5 Conclusion

In this paper, we considered the problem of simplifying the visibility polygon of an observer inside a planar scene. This problem has many applications in computer graphics, games, robotics, path planning, and GIS. We first defined observer-dependent error functions for both distance-distortion and area-distortion simplification metrics. Our goal in defining these simplification criteria was to consider the position of the observer as an effective factor in real simplification.

For our definitions of the observer-dependent simplification, we described algorithms to simplify the visibility polygon of a point observer, 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. Although, the new criteria are mathematically more complicated, but, our simplification algorithms are as efficient as the best current simplification algorithms that do not take the observer position into account.

The notion of observer-dependent simplification is pretty new and is applicable in real applications and there are many directions in extending, improving and implementing the methods described here. Extending to three dimensional space, line segment observers and implementation issues are examples of the next open directions.

References

1. Abam, M.A., de Berg, M., Hachenberger, P., Zarei, A.: Streaming algorithms for line simplification. Discrete Comput. Geom. **43**(3), 497–515 (2010)
2. Agarwal, P.K., Yu, H.: A space-optimal data-stream algorithm for coresets in the plane. In: Proc. 23th ACM Symposium on Computational Geometry (SoCG), pp. 1–10 (2007)
3. Agarwal, P.K., Varadarajan, K.R.: Efficient algorithms for approximating polygonal chains. Discrete Comput. Geom. **23**(2), 273–291 (2000)
4. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. Int. J. Comput. Geom. Appl. **5**, 75–91 (1995)
5. Bose, P., Cabello, S., Cheong, O., Gudmundsson, J., Kreveld, M., Speckmann, B.: Area-preserving approximations of polygonal paths. J. Discrete Algorithms **4**, 554–566 (2006)
6. Brodal, G.S., Jacob, R.: Dynamic planar convex hull. In: Proc. 43rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 617–626 (2002)

7. Buzer, L.: Optimal simplification of polygonal chain for rendering. In: Proc. 23rd ACM Symposium on Computational Geometry (SoCG), pp. 168–174 (2007)
8. Chan, W.S., Chin, F.: Approximation of polygonal curves with minimum number of line segments. In: Proc. 3rd Ann. Int. Sym. on Alg. and Comp. (ISAAC). LNCS, vol. 650, pp. 378–387 (1992)
9. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Can. Cartogr. **10**, 112–122 (1973)
10. Godau, M.: A natural metric for curves: computing the distance for polygonal chains and approximation algorithms. In: Proc. 8th Ann. Symp. on Theo. Aspects of Computer Science (STACS), pp. 127–136 (1991)
11. Goodrich, M.T.: Efficient piecewise-linear function approximation using the uniform metric. Discrete Comput. Geom. **14**, 445–462 (1995)
12. Guibas, L.J., Hershberger, J.E., Mitchell, J.S.B., Snoeyink, J.S.: Approximating polygons and subdivisions with minimum link paths. Int. J. Comput. Geom. Appl. **3**, 383–415 (1993)
13. Hakimi, S.L., Schmeichel, E.F.: Fitting polygonal functions to a set of points in the plane. CVGIP: Graph. Models Image Process. **53**, 132–136 (1991)
14. Hershberger, J., Snoeyink, J.: An $O(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification. In: Proc. 10th ACM Symposium on Computational Geometry (SoCG), pp. 383–384 (1994)
15. Hershberger, J., Snoeyink, J.: Computing minimum length paths of a given homotopy class. Comput. Geom.: Theory Appl. **4**(2), 63–97 (1994)
16. Imai, H., Iri, M.: An optimal algorithm for approximating a piecewise linear function. J. Inf. Process. **9**(3), 159–162 (1986)
17. Imai, H., Iri, M.: Polygonal approximations of a curve-formulations and algorithms. In: Computational Morphology, pp. 71–86. North-Holland, Amsterdam (1988)
18. McMaster, R.B.: A quantitative analysis of mathematical measures in linear simplification. Ph.D. Thesis, Dept. of Geography and Meteorology, University of Kansas, Lawrence, Kansas (1983)
19. McMaster, R.B.: The geometric properties of numerical generalization. Geogr. Anal. **19**(4), 330–346 (1987)
20. Melkman, A.: On-line construction of the convex hull of a simple polyline. Inf. Process. Lett. **25**, 11–12 (1987)
21. Melkman, A., ORourke, J.: On polygonal chain approximation. In: Computational Morphology, pp. 87–95. North-Holland, Amsterdam (1988)
22. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer, New York (1985)
23. Veregin, H.: Line simplification, geometric distortion, and positional error. Cartographica **36**(1), 25–39 (1999)
24. Visvalingam, M., Whyatt, J.D.: Line generalisation by repeated elimination of points. Cartogr. J. **30**(1), 46–51 (1993)