An Improved Algorithm for Reconstructing a Simple Polygon from the Visibility Angles

Danny Z. Chen† Haitao Wang‡

Abstract

In this paper, we study the following problem of reconstructing a simple polygon: Given a cyclically ordered vertex sequence of an unknown simple polygon $P$ of $n$ vertices and, for each vertex $v$ of $P$, the sequence of angles defined by all the visible vertices of $v$ in $P$, reconstruct the polygon $P$ (up to similarity). An $O(n^3 \log n)$ time algorithm has been proposed for this problem. We present an improved algorithm with running time $O(n^2)$, based on new observations on the geometric structures of the problem. Since the input size (i.e., the total number of input visibility angles) is $\Theta(n^2)$ in the worst case, our algorithm is worst-case optimal.

1 Introduction

In this paper, we study the problem of reconstructing a simple polygon $P$ from the visibility angles measured at the vertices of $P$ and from the cyclically ordered vertices of $P$ along its boundary. Precisely, for an unknown simple polygon $P$ of $n$ vertices, suppose we are given (1) the vertices ordered counterclockwise (CCW) along the boundary of $P$, and (2) for each vertex $v$ of $P$, the angles between any two adjacent rays emanating from $v$ to the vertices of $P$ that are visible to $v$ such that these angles are in the CCW order as seen around $v$, beginning at the CCW neighboring vertex of $v$ on the boundary of $P$ (e.g., see Fig. 1). A vertex $u$ of $P$ is visible to a vertex $v$ of $P$ if the line segment connecting $u$ and $v$ lies entirely in $P$. The objective of the problem is to reconstruct the simple polygon $P$ (up to similarity) that fits all the given angles. We call this problem the polygon reconstruction problem from angles, or PRA for short. Figure 2 gives an example.

The PRA problem has been studied by Disser, Mihal’ak, and Widmayer [6], who showed that the solution polygon for the input is unique (up to similarity) and gave an $O(n^3 \log n)$ time algorithm for reconstructing such a polygon. Using the input, their algorithm first constructs the visibility graph $G$ of $P$ and subsequently reconstructs the polygon $P$. As shown in [6], once $G$ is known, the polygon $P$ can be obtained efficiently (e.g., in $O(n^2)$ time) with the help of the angle data and the CCW ordered vertex sequence of $P$.

Given a visibility graph $G$, the problem of determining whether there is a polygon $P$ that has $G$ as its visibility graph is called the visibility graph recognition problem, and the problem of actually constructing such a polygon $P$ is called the visibility graph reconstruction problem. Note that the general visibility graph recognition and reconstruction problems are long-standing open problems with only partial results known (e.g., see [1] for a short survey). Everett [8] showed that the visibility graph reconstruction problem is in PSPACE, but no better upper bound on the complexity of either problem is known. In our problem setting, we have the angle data information and the ordered vertex list of $P$; thus $P$ can be constructed efficiently after knowing $G$.

*This research was supported in part by NSF under Grant CCF-0916606.
†Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: {dchen, hwang6}@nd.edu.
‡Corresponding author.
Hence, the major part of the algorithm in [6] is dedicated to constructing the visibility graph $G$ of $P$. As indicated in [6], the key difficulty is that the vertices in this problem setting have no recognizable labels, e.g., the angle measurement at a vertex $v$ gives angles between visible vertices to $v$ but does not identify these visible vertices globally. The authors in [6] also showed that some natural greedy approaches do not seem to work. An $O(n^3 \log n)$ time algorithm for constructing $G$ is given in [6]. The algorithm, called the triangle witness algorithm, is based on the following observation: Suppose we wish to determine whether a vertex $v_i$ is visible to another vertex $v_j$; then $v_i$ is visible to $v_j$ if and only if there is a vertex $v_l$ on the portion of the boundary of $P$ from $v_{i+1}$ to $v_{j-1}$ in the CCW order such that $v_l$ is visible to both $v_i$ and $v_j$ and the triangle formed by the three vertices $v_i, v_j$, and $v_l$ does not intersect the boundary of $P$ except at these three vertices (such a vertex $v_l$ is called a triangle witness vertex).

In this paper, based on the triangle witness algorithm [6], by exploiting some new geometric properties, we give an improved algorithm with a running time of $O(n^2)$. The improvement is due to two key observations. First, in the triangle witness algorithm [6], to determine whether a vertex $v_i$ is visible to another vertex $v_j$, the algorithm needs to determine whether there exists a triangle witness vertex along the boundary of $P$ from $v_{i+1}$ to $v_{j-1}$ in the CCW order; to this end, the algorithm checks every vertex in that boundary portion of $P$. We observe that it suffices to check only one particular vertex in that boundary portion. This removes an $O(n)$ factor from the running time of the triangle witness algorithm [6]. Second, some basic operations in the triangle witness algorithm [6] take $O(\log n)$ time each; by utilizing certain different data structures, our new algorithm can handle each of those basic operations in constant time. This removes another $O(\log n)$ factor from the running time. Note that since the input size is $\Theta(n^2)$ in the worst case (e.g., the total number of all visibility angles), our algorithm is worst-case optimal.

As shown in [6], if only the angle measurements are given, i.e., the ordered vertices along the boundary of $P$ are unknown, then the information is not sufficient for reconstructing $P$. In other words, it may be possible to compute several simple polygons that are not similar but all fit the given measured angles (see [6] for an example).
1.1 Related Work

The problems of reconstructing geometric objects based on measurement data have been studied extensively (e.g., [2, 3, 12, 14, 15]). As discussed above, the general visibility graph recognition and reconstruction problems are in PSPACE [8] and no better complexity upper bound is known so far (e.g., see [10]). Yet, some results have been given for certain special polygons. For example, Everett and Corneil [9] characterized the visibility graphs of spiral polygons and gave a linear time reconstruction algorithm. Choi, Shin, and Chwa [4], and Colley, Lubiw, and Spinrad [5] characterized and recognized the visibility graphs of funnel-shaped polygons.

By adding extra information, some versions of the problems become more tractable. O’Rourke and Streinu [13] considered the vertex-edge visibility graph that includes edge-to-edge visibility information. Wismath [17] introduced the stab graphs which are also an extended visibility structure and showed how parallel line segments can be efficiently reconstructed from it. Snoeyink [15] proved that a unique simple polygon (up to similarity) can be determined by the interior angles at its vertices and the cross-ratios of the diagonals of any given triangulation, where the cross-ratio of a diagonal is the product of the ratios of edge lengths for the two adjacent triangles. Jackson and Wismath [12] studied the reconstruction of orthogonal polygons from horizontal and vertical visibility information and gave an $O(n \log n)$ time reconstruction algorithm. Biedl, Durocher, and Snoeyink [2] considered the problem of reconstructing the two-dimensional floor plan of a polygonal room using different types of scanned data, and proposed several problem models. Sidlesky, Barequet, and Gotsman [14] studied the problem of reconstructing a planar polygon from its intersections with a collection of arbitrarily-oriented “cutting” lines.

Reconstructing a simple polygon from angle data was first considered by Bilò et al. [3], who aimed to understand what kinds of sensorial capabilities are sufficient for a robot moving inside an unknown polygon to reconstruct the visibility graph of the polygon. It was shown in [3] that if the robot is equipped with a compass to measure the angle between any two vertices that are currently visible to the robot and also has the ability to know where it came from when moving from vertex to vertex, then the visibility graph of the polygon can be uniquely reconstructed. Reconstruction and exploration of environments by robots in other problem settings have also been studied (e.g., see [7, 11, 16]).

The rest of this paper is organized as follows. In Section 2 we give the problem definitions in detail and introduce some notations and basic observations. To be self-contained, in Section 3 we briefly review the triangle witness algorithm given in [6]. We then present our improved algorithm in Section 4.

2 Preliminaries

In this section, we define the PRA problem in detail and introduce the needed notations and terminology. For ease of discussion and comparison, some of our notations follow those in [6].

Let $P$ be a simple polygon of $n$ vertices $v_0, v_1, \ldots, v_{n-1}$ in the CCW order along $P$’s boundary. Denote by $G = (V, E)$ the visibility graph of $P$, where $V$ consists of all vertices of $P$ and for any two distinct vertices $v_i$ and $v_j$, $E$ contains an edge $e(v_i, v_j)$ connecting $v_i$ and $v_j$ if and only if $v_i$ is visible to $v_j$ inside $P$. In this paper, the indices of all $v_i$’s are taken as congruent modulo $n$, i.e., if $n \leq i + j \leq 2n - 1$, then $v_{i+j}$ is the same vertex as $v_l$, where $l = i + j - n$ (or $l = (i + j) \mod n$); similarly, if $-n \leq i - j < 0$, then $v_{i-j}$ is the same vertex as $v_l$, where $l = i - j + n$. For each $v_i \in V$, denote by $deg(v_i)$ its degree in the visibility graph $G$, and denote
by \( \text{vis}(v_i) = (\text{vis}_1(v_i), \text{vis}_2(v_i), \ldots, \text{vis}_{\deg(v_i)}(v_i)) \) the sequence of vertices in \( P \) visible to \( v_i \) from \( v_{i+1} \) to \( v_{i-1} \) ordered CCW around \( v_i \). We refer to \( \text{vis}(v_i) \) as \( v_i \)'s visibility angle sequence. Note that since both \( v_{i-1} \) and \( v_{i+1} \) are visible to \( v_i \), \( \text{vis}_1(v_i) = v_{i+1} \) and \( \text{vis}_{\deg(v_i)}(v_i) = v_{i-1} \). For any two vertices \( v_i, v_j \) in \( P \), let \( ch(v_i, v_j) \) denote the sequence \( (v_i, v_{i+1}, \ldots, v_j) \) of the vertices ordered CCW along the boundary of \( P \) from \( v_i \) to \( v_j \). We refer to \( ch(v_i, v_j) \) as a chain. Let \( |ch(v_i, v_j)| \) denote the number of vertices of \( P \) in the chain \( ch(v_i, v_j) \).

For any two distinct vertices \( v_i, v_j \in P \), let \( \rho(v_i, v_j) \) be the ray emanating from \( v_i \) and going towards \( v_j \). For any three vertices \( v, v_i, v_j \in P \), denote by \( \ang_i(v, v_j) \) the CCW angle defined by rotating \( \rho(v, v_i) \) around \( v \) to \( \rho(v, v_j) \) (\( v_i \) or \( v_j \) need not be visible to \( v \)). Note that the values of all angles we use in this paper are in \([0, 2\pi)\). For any vertex \( v \in P \) and \( 1 \leq i < j \leq \deg(v) \), let \( \ang_v(i, j) \) be \( \ang_v(\text{vis}_i(v), \text{vis}_j(v)) \).

The PRA problem can then be re-stated as follows: Given a sequence of all vertices \( v_0, v_1, \ldots, v_{n-1} \) of an unknown simple polygon \( P \) in the CCW order along \( P \)'s boundary, and the angles \( \ang_v(i, i+1) \) for each vertex \( v \) of \( P \) with \( 1 \leq i < \deg(v) \), we seek to reconstruct \( P \) (up to similarity) to fit all the given angles. Without loss of generality, we assume that no three distinct vertices of \( P \) are collinear.

It is easy to see that after \( O(n^2) \) time preprocessing, for any \( v \in V \) and any \( 1 \leq i < j \leq \deg(v) \), the angle \( \ang_v(i, j) \) can be obtained in constant time. In the following discussion, we assume that this preprocessing has already been done. Sometimes we (loosely) say that these angles are given as input.

The algorithm given in [6] does not construct \( P \) directly. Instead, the algorithm first computes its visibility graph \( G = (V, E) \). As mentioned earlier, after knowing \( G, P \) can be reconstructed efficiently with the help of the angle data and the CCW ordered vertex sequence of \( P \). The algorithm for constructing \( G \) in [6] is called the triangle witness algorithm, which will be briefly reviewed in Section 3. Since \( V \) consists of all vertices of \( P \), the problem of constructing \( G \) is equivalent to constructing its edge set \( E \), i.e., for any two distinct vertices \( v_i, v_j \in P \), determine whether there is an edge \( e(v_i, v_j) \in E \) connecting \( v_i \) and \( v_j \) (in other words, determining whether \( v_i \) is visible to \( v_j \) inside \( P \)).

To discuss the involved algorithms, we need one more definition. For any two vertices \( v_i, v_j \in P \) with \( |ch(v_{i+1}, v_{j-1})| \geq 1 \), suppose a vertex \( v_l \in ch(v_{i+1}, v_{j-1}) \) is visible to both \( v_i \) and \( v_j \); then we let \( v'_j \) be the first visible vertex to \( v_i \) on the chain \( ch(v_j, v_i) \) and let \( v'_l \) be the last visible vertex to \( v_i \) on the chain \( ch(v_i, v_l) \) (e.g., see Fig. 3). Intuitively, imagine that we rotate a ray from \( \rho(v_l, v_i) \) around \( v_i \) counterclockwise; then the first vertex on the chain \( ch(v_j, v_i) \) hit by the rotating ray is \( v'_j \). Similarly, if we rotate a ray from \( \rho(v_j, v_l) \) around \( v_j \) clockwise, then the first vertex on the chain \( ch(v_j, v_i) \) hit by the rotating ray is \( v'_l \). Note that if \( v_i \) is visible to \( v_j \), then \( v'_j = v_j \) and \( v'_l = v_i \). We denote by \( \ang_{v_i}(v_l, v_j) \) the angle \( \ang_{v_i}(v_l, v'_j) \) and denote by \( \ang_{v_i}(v_j, v_l) \) the angle \( \ang_{v_j}(v'_i, v_l) \). It should be pointed out that for ease of understanding this paper, the above statement of defining \( \ang^{\uparrow} \) is different from that in [6] but they refer to the same angles in the algorithm. The motivation for defining \( \ang^{\uparrow} \) will be clear after discussing the following lemma, which has been proved in [6].

**Lemma 1** [6] For any two vertices \( v_i, v_j \in P \) with \( |ch(v_{i+1}, v_{j-1})| \geq 1 \), \( v_i \) is visible to \( v_j \) if and only if there exists a vertex \( v_l \) on \( ch(v_{i+1}, v_{j-1}) \) such that \( v_l \) is visible to both \( v_i \) and \( v_j \) and \( \ang_{v_i}(v_l, v_j) + \ang_{v_j}(v_l, v_i) + \ang_{v_i}(v_j, v_l) = \pi \).

Since the above lemma is also critical to our improved algorithm in Section 4, we sketch the proof of the lemma below.
Figure 3: Illustrating the definitions of $v'_i$ and $v'_j$, and $\angle_{v_i}(v_i, v_j)$ and $\angle_{v_i}(v_i, v_k)$ which are the angles $\angle_{v_i}(v_i, v'_j)$ and $\angle_{v_i}(v'_i, v_i)$, respectively.

For any two vertices $v_i, v_j \in P$ with $|ch(v_{i+1}, v_{j-1})| \geq 1$, if $v_i$ is visible to $v_j$, then it is not difficult to see that there must exist a vertex $v_t \in ch(v_{i+1}, v_{j-1})$ that is visible to both $v_i$ and $v_j$. Since the three vertices $v_i, v_j,$ and $v_t$ are mutually visible to each other, it is clear that the triangle formed by these three vertices does not intersect the boundary of $P$ except at the three vertices, implying that $\angle_{v_i}(v_t, v_j) + \angle_{v_j}(v_i, v_j) + \angle_{v_j}(v_i, v_t) = \angle_{v_i}(v_i, v_j) + \angle_{v_i}(v_i, v_j) + \angle_{v_i}(v_i, v_t) = \pi$. Such a vertex $v_t$ is called a triangle witness of the edge $e(v_i, v_j)$ in $E$.

Suppose $v_i$ is not visible to $v_j$; then it is possible that there does not exist a vertex $v_t \in ch(v_{i+1}, v_{j-1})$ which is visible to both $v_i$ and $v_j$. In fact, if there exists no vertex $v_t \in ch(v_{i+1}, v_{j-1})$ that is visible to both $v_i$ and $v_j$, then $v_i$ cannot be visible to $v_j$. Hence in the following, we assume that such a vertex $v_t$ exists, i.e., a vertex $v_t \in ch(v_{i+1}, v_{j-1})$ is visible to both $v_i$ and $v_j$ (yet $v_t$ is not visible to $v_j$). Note that $v_t$ is not collinear with $v_i$ and $v_j$. If $\angle_{v_t}(v_j, v_i) > \pi$ (i.e., the chain $ch(v_i, v_j)$ is part of the boundary of $P$ that blocks the visibility between $v_i$ and $v_j$), then the lemma obviously holds. Otherwise, $\angle_{v_t}(v_j, v_i) < \pi$, and in this case, the visibility between $v_i$ and $v_j$ is blocked by the chain $ch(v_i, v_j)$. Since $v_i$ is not visible to $v_j$, for any choice of such a vertex $v_t$, the angle $\angle_{v_t}(v_i, v_j)$ is not given by $v_i$’s visibility angle sequence $vis(v_i)$. The “closest approximation” for $\angle_{v_i}(v_j, v_i)$ in this case is determined by a vertex $v$ on the chain $ch(v_i, v_j)$ such that $v$ becomes $v_j$ if and only if $v_j$ is visible to $v_i$. As in the definition of $\angle_{v_i}(v_i, v_j)$, the vertex $v_j'$ is such a vertex $v$, i.e., the first visible vertex to $v_i$ on the chain $ch(v_j, v_i)$. Similarly, the vertex $v_i$ in $\angle_{v_i}(v_i, v_t)$ is “replaced” in the definition of $\angle_{v_t}(v_i, v_i)$ by $v_j'$, i.e., the last visible vertex to $v_i$ on $ch(v_j, v_i)$. Clearly, when $v_j$ is not visible to $v_i$, it must hold that $\angle_{v_i}(v_i, v_j') < \angle_{v_i}(v_i, v_t)$ and $\angle_{v_i}(v_i, v_j) < \angle_{v_t}(v_i, v_i)$. Therefore, $\angle_{v_i}(v_t, v_j) + \angle_{v_j}(v_i, v_j) + \angle_{v_j}(v_i, v_t) = \angle_{v_i}(v_i, v_j') + \angle_{v_j}(v_i, v_t) + \angle_{v_i}(v_j, v_j) + \angle_{v_i}(v_i, v_t) = \pi$. Lemma 1 thus follows.

3 The Triangle Witness Algorithm

In this section, we briefly review the triangle witness algorithm in [6] that constructs the visibility graph $G = (V, E)$ of the unknown simple polygon $P$.

The triangle witness algorithm is based on Lemma 1. The algorithm has $\lceil n/2 \rceil$ iterations. In the $k$-th iteration ($1 \leq k \leq \lceil n/2 \rceil$), the algorithm checks, for each $i = 0, 1, \ldots, n - 1$, whether $v_i$ is visible to $v_{i+k}$. After all iterations, the edge set $E$ can be obtained. To this end, the algorithm maintains two maps $F$ and $B$: $F[v_i][v_j] = t$ if $v_j$ is identified as the $t$-th visible vertex to $v_i$ in the CCW order, i.e., $v_j = vis_t(v_i)$; the definition of $B$ is the same as $F$. During the algorithm, $F$ will be filled in the CCW order and $B$ will be filled in the clockwise (CW) order. When the algorithm finishes, for each $v_i$, $F[v_i]$ will have all visible vertices to $v_i$ on the chain $ch(v_{i+1}, v_{[n/2]})$. 


while $B[v_i]$ will have all visible vertices to $v_i$ on the chain $ch(v_{i\lceil n/2\rceil}, v_{i-1})$. Thus, $F[v_i]$ and $B[v_i]$ together contain all visible vertices of $P$ to $v_i$. For ease of description, we also treat $F[v_i]$ and $B[v_i]$ as sets, e.g., $v_i \in F[v_i]$ means that there is an entry $F[v_i][v_i]$ and $|F[v_i]|$ means the number of entries in the current $F[v_i]$.

Initially, when $k = 1$, since every vertex is visible to its two neighbors along the boundary of $P$, we have $F[v_i][v_{i+1}] = 1$ and $B[v_i][v_{i-1}] = \deg(v_i)$ for each $v_i$. In the $k$-th iteration, we determine for each $v_i$, whether $v_i$ is visible to $v_{i+k}$. Below, we let $j = i + k$. Note that $|F[v_i]| + 1$ is the index of the first visible vertex to $v_i$ in the CCW order that is not yet identified; similarly, $(\deg(v_j) - |B[v_j]|)$ is the index of the first visible vertex to $v_j$ in the CW order that is not yet identified. If $v_i$ is visible to $v_j$, then we know that $v_j$ is the $([F[v_i]| + 1)$-th visible vertex to $v_i$ and $v_i$ is the $(\deg(v_j) - |B[v_j]|)$-th visible vertex to $v_j$, and thus we set $F[v_i][v_j] = |F[v_i]| + 1$ and $B[v_j][v_i] = \deg(v_j) - |B[v_j]|$. If $v_i$ is not visible to $v_j$, then we do nothing.

It remains to discuss how to determine whether $v_i$ is visible to $v_j$. According to Lemma 4, we need to determine whether there exists a triangle witness vertex $v_i$ in the chain $ch(v_{i+1}, v_{j-1})$, i.e., $v_i$ is visible to both $v_i$ and $v_j$ and $\angle_{v_i}(v_i, v_j) + \angle_{v_j}(v_i, v_j) + \angle_{v_j}(v_j, v_i) = \pi$. To this end, the algorithm checks every vertex $v_i$ in $ch(v_{i+1}, v_{j-1})$. For each $v_i$, the algorithm first determines whether $v_i$ is visible to both $v_i$ and $v_j$, by checking whether there is an entry for $v_i$ in $F[v_i]$ and in $B[v_j]$. The algorithm utilizes balanced binary search trees to represent $F$ and $B$, and thus checking whether there is an entry for $v_i$ in $F[v_i]$ and in $B[j]$ can be done in $O(\log n)$ time. If $v_i$ is visible to both $v_i$ and $v_j$, then the next step is to determine whether $\angle_{v_i}(v_i, v_j) + \angle_{v_j}(v_i, v_j) + \angle_{v_j}(v_j, v_i) = \pi$. It is easy to know that $\angle_{v_i}(v_j, v_i)$ is $\angle_{v_i}(F[v_i][v_j], B[v_i][v_j])$, which can be found readily from the input. To obtain $\angle_{v_i}(v_i, v_j)$, we claim that it is the angle $\angle_{v_i}(F[v_i][v_i], |F[v_i]| + 1)$. Indeed, observe that all visible vertices in the chain $ch(v_{i+1}, v_{j-1})$ are in the current $F[v_i]$. As explained above, $|F[v_i]| + 1$ is the index of the first visible vertex to $v_i$ in the CCW order that has not yet been identified, which is the first visible vertex to $v_i$ in the chain $ch(v_j, v_i)$, i.e., the vertex $v_j'$. Thus, $\angle_{v_i}(F[v_i][v_i], |F[v_i]| + 1)$ is $\angle_{v_i}(v_i, v_j)$, which is $\angle_{v_i}(v_i, v_j)$. Similarly, $\angle_{v_i}(v_i, v_i)$ is the angle $\angle_{v_j}(deg(v_j) - |B[v_j]|, B[v_j][v_j])$. Note that both the angles $\angle_{v_i}(F[v_i][v_i], |F[v_i]| + 1)$ and $\angle_{v_j}(deg(v_j) - |B[v_j]|, B[v_j][v_j])$ are known. Algorithm 2 in the Appendix summarizes the whole algorithm 6.

To analyze the running time of the above triangle witness algorithm, note that it has $\lceil n/2 \rceil$ iterations. In each iteration, the algorithm checks whether $v_i$ is visible to $v_{i+k}$ for each $0 \leq i \leq n-1$. For each $v_i$, the algorithm checks every $v_i$ for $i+1 \leq l \leq i+k-1$, i.e., in the chain $ch(v_{i+1}, v_{i+k-1})$. For each such $v_i$, the algorithm takes $O(\log n)$ time as it uses balanced binary search trees to represent the two maps $F$ and $B$. In summary, the overall running time of the triangle witness algorithm in 6 is $O(n^3 \log n)$.

4 An Improved Triangle Witness Algorithm

In this section, we present an improved solution over the triangle witness algorithm in 6 sketched in Section 3. Our improved algorithm runs in $O(n^2)$ time. Since the input size (e.g., the total number of visibility angles) is $\Theta(n^2)$ in the worst case, our improved algorithm is worst-case optimal. Our new algorithm follows the high-level scheme of the triangle witness algorithm in 6, and thus we call it the improved triangle witness algorithm.

As in 6, the new algorithm also has $\lceil n/2 \rceil$ iterations, and in each iteration, we determine whether $v_i$ is visible to $v_{i+k}$ for each $0 \leq i \leq n-1$. For every pair of vertices $v_i$ and $v_{i+k}$, let $j = i + k$. To determine whether $v_i$ is visible to $v_j$, the triangle witness algorithm 6 checks each vertex $v_i$ in the chain $ch(v_{i+1}, v_{j-1})$ to see whether there exists a triangle witness vertex. In our
new algorithm, instead, we claim that we need to check only on e particular vertex, \( v \), if and only if the following lemma.

**Lemma 2** The vertex \( v_i \) is visible to \( v_j \) if and only if the vertex \( vis|_{F[v_i]}(v_i) \) is a triangle witness vertex for \( v_i \) and \( v_j \).

**Proof:** Let \( v' \) denote the vertex \( vis|_{F[v_i]}(v_i) \), which is the last visible vertex to \( v_i \) in the chain \( ch(v_{i+1}, v_{j-1}) \) in the CCW order. Recall that \( v' \) being a triangle witness vertex for \( v_i \) and \( v_j \) is equivalent to saying that \( v' \) is visible to both \( v_i \) and \( v_j \) and \( \angle_{v_i}^+(v', v_j) + \angle_{v_j}^+(v_i, v') + \angle_{v_j}^+(v_j, v_i) = \pi \).

If \( v_i \) is visible to \( v_j \), then we prove below that \( v' \) is a triangle witness vertex for \( v_i \) and \( v_j \), i.e., we prove that \( v' \) is visible to both \( v_i \) and \( v_j \) and \( \angle_{v_i}^+(v', v_j) + \angle_{v_j}^+(v_i, v') + \angle_{v_j}^+(v_j, v_i) = \pi \). Refer to Fig. 4 for an example. Let \( P(v_i) \) denote the subpolygon of \( P \) that is visible to the vertex \( v_i \). Usually, \( P(v_i) \) is called the *visibility polygon* of \( v_i \) and it is well-known that \( P(v_i) \) is a star-shaped polygon with \( v_i \) as a kernel point (e.g., see [1]). Figure 5 illustrates \( P(v_i) \). Since \( v' \) is the last visible vertex to \( v_i \) in \( ch(v_{i+1}, v_{j-1}) \) and \( v_j \) is visible to \( v_i \), we claim that \( v_j \) is also visible to \( v' \). Indeed, since both \( v' \) and \( v_j \) are visible to \( v_i \), \( v' \) and \( v_j \) are both on the boundary of the visibility polygon \( P(v_i) \). Let \( P'(v_i) \) denote the portion of the boundary of \( P(v_i) \) from \( v' \) to \( v_j \) counterclockwise (see Fig. 5). We prove below that \( P'(v_i) \) does not contain any vertex of \( P \) except \( v' \) and \( v_j \). First, \( P'(v_i) \) cannot contain any vertex in the chain \( ch(v_{i+1}, v_{i-1}) \) (otherwise, \( v_j \) would not be visible to \( v_i \)). Let the index of the vertex \( v' \) be \( l \), i.e., \( v' = v_l \). Similarly, \( P'(v_i) \) cannot contain any vertex in the chain \( ch(v_{i+1}, v_{l-1}) \) (otherwise, \( v' \) would not be visible to \( v_i \)). Finally, \( P'(v_i) \) cannot contain any vertex in the chain \( ch(v_l+1, v_{j-1}) \), since otherwise \( v' \) would not be the last visible vertex to \( v_i \) in \( ch(v_{i+1}, v_{j-1}) \). Thus, \( P'(v_i) \) does not contain any vertex of \( P \) except \( v' \) and \( v_j \). Therefore, the region bounded by \( P'(v_i) \),
the line segment connecting $v_i$ and $v_j$, and the line segment connecting $v_i$ and $v'$ must be convex (in fact, it is always a triangle) and this region is entirely contained in $P$. This implies that $v_j$ is visible to $v'$. Hence, the three vertices $v_i, v_j$, and $v'$ are mutually visible to each other, and we have $\angle_{v_i}(v_i,v_j) + \angle_{v_i}(v_i,v_j) + \angle_{v_i}(v_i,v_j) = \angle_{v_i}(v_i,v_j) + \angle_{v_i}(v_i,v_j) + \angle_{v_i}(v_i,v_j) = \pi$.

On the other hand, if $v'$ is a triangle witness vertex for $v_i$ and $v_j$, then by Lemma 4, the vertex $v_i$ is visible to $v_j$. The lemma thus follows. \hfill \Box

By Lemma 2 to determine whether $v_i$ is visible to $v_j$, instead of checking every vertex in $ch(v_i+1, v_j-1)$, we need to consider only the vertex $vis_{F[v_i]}(v_i)$ in the current set $F[v_i]$. Hence, Lemma 2 immediately reduces the running time of the triangle witness algorithm by an $O(n)$ factor. The other $O(n)$ factor improvement is due to a new way of defining and representing the maps $F$ and $B$, as elaborated below. In the following discussion, let $v'$ be the vertex $vis_{F[v_i]}(v_i)$.

In our new algorithm, to determine whether $v_i$ is visible to $v_j$, we check whether $v'$ is a triangle witness vertex for $v_i$ and $v_j$. To this end, we already know that $v'$ is visible to $v_i$, but we still need to check whether $v'$ is visible to $v_j$. In the previous triangle witness algorithm [6], this step is performed in $O(n)$ time by representing $F$ and $B$ using balanced binary search trees. In our new algorithm, we handle this step in $O(1)$ time, by redefining $F$ and $B$ and using a new way to represent them.

We redefine $F$ as follows: $F[v_i][v_j] = t$ if $v_j$ is the $t$-th visible vertex to $v_i$ in the CCW order; if $v_j$ is not visible to $v_i$ or $v_j$ has not yet been identified, then $F[v_i][v_j] = 0$. For convenience, we let $F[v_i]$ = $F[v_i][v_i+1, v_i+2, \ldots, v_i-1]$. Thus, in our new definition, the size of $F[v_i]$ is fixed throughout the algorithm, i.e., $|F[v_i]|$ is always $n-1$. In addition, for each $v_i$, the new algorithm maintains two variables $L_i$ and $I_i$ for $F[v_i]$, where $L_i$ is the number of non-zero entries in the current $F[v_i]$, which is also the number of visible vertices to $v_i$ that have been identified (i.e., the number of visible vertices to $v_i$ in the chain $ch(v_i+1, v_i+k-1)$) up to the $k$-th iteration, and $I_i$ is the index of the last non-zero entry in the current $F[v_i]$, i.e., $I_i$ is the last visible vertex to $v_i$ in the chain $ch(v_i+1, v_i+k-1)$ in the CCW order. Similarly, we redefine $B$ in the same way as $F$, i.e., for each $v_i$, $B[v_i] = B[v_i][v_i+1, v_i+2, \ldots, v_i-1]$ and $B[v_i][v_j] = t$ if $v_j$ is the $t$-th visible vertex to $v_i$ in the CCW order. Further, for each $v_i$, we also maintain two variables $L'_i$ and $I'_i$ for $F[v_i]$, where $L'_i$ is the number of non-zero entries in the current $B[v_i]$, which is also the number of visible vertices to $v_i$ in the chain $ch(v_i-k+1, v_i-1)$ (up to the $k$-th iteration), and $I'_i$ is the index of the first non-zero entry in the current $B[v_i]$, i.e., $I'_i$ is the first visible vertex to $v_i$ in the chain $ch(v_i-k+1, v_i-1)$ in the CCW order. During the algorithm, the array $F[v_i]$ will be filled in the CCW order, i.e., from the first entry $F[v_i][v_i+1]$ to the end while the array $B[v_i]$ will be filled in the CW order, i.e., from the last entry $B[v_i][v_i-1]$ to the beginning. When the algorithm finishes, $F[v_i]$ will contain all the visible vertices to $v_i$ in the chain $ch(v_i+1, v_i+[n/2])$, and thus only the entries of the first half of
Algorithm 1: The Improved Triangle Witness Algorithm

Input: \( n, (v_1, v_2, \ldots, v_n), \deg(v_i) \) and the angles \( \angle_\alpha(\cdot, \cdot) \) for each \( 0 \leq i \leq n - 1 \).

Output: The edge set \( E \) of the visibility graph \( G \) for the target polygon \( P \).

/* All indices below are understood modulo \( n \). */
\[
E \leftarrow \emptyset;
\]

for \( i \leftarrow 0 \) to \( n - 1 \) do
\[
F[v_i][v_{i+1}, v_{i+2}, \ldots, v_{i-1}] \leftarrow 0;
B[v_i][v_{i+1}, v_{i+2}, \ldots, v_{i-1}] \leftarrow 0;
\]

for \( i \leftarrow 0 \) to \( n - 1 \) do
\[
E \leftarrow E \cup \{e(v_i, v_{i+1})\};
F[v_i][v_{i+1}] \leftarrow 1;
B[v_i][v_{i-1}] \leftarrow \deg(v_i);
L_i \leftarrow 1, I_i \leftarrow v_{i+1}, L'_i \leftarrow 1, I'_i \leftarrow v_{i-1};
\]

for \( k \leftarrow 2 \) to \( \lceil n/2 \rceil \) do
\[
\text{for } i \leftarrow 0 \text{ to } n - 1 \text{ do }
\]
\[
j \leftarrow i + k;
\]
\[
v' \leftarrow I_i;
\]
\[
\text{if } B[v_j][v'] > 0 \text{ then }
\]
\[
\angle^\uparrow_{v_i}(v', v_j) \leftarrow \angle_{v_i}(F[v_i][v'], L_i + 1);
\]
\[
\angle^\downarrow_{v_i}(v_i, v') \leftarrow \angle_{v_i}(\deg(v_i) - L'_i, B[v_j][v']);
\]
\[
\angle_{v'}(v_j, v_i) \leftarrow \angle_{v'}(F[v'][v_j], B[v'][v_i]);
\]
\[
\text{if } \angle^\uparrow_{v_i}(v', v_j) + \angle^\downarrow_{v_i}(v_i, v') + \angle_{v'}(v_j, v_i) = \pi \text{ then }
\]
\[
E \leftarrow E \cup \{e(v_i, v_j)\};
F[v_i][v_j] \leftarrow L_i + 1;
B[v_j][v_i] \leftarrow \deg(v_j) - L'_j;
L_i \leftarrow L_i + 1, L'_i \leftarrow L'_i + 1, I_i \leftarrow v_j, I'_i \leftarrow v_i;
\]

\( F[v_i] \) are possibly filled with non-zero values. Similarly, only the entries of the second half of \( B[v_i] \) are possibly filled with non-zero values. Below, we discuss the implementation details of our new algorithm, which is summarized in Algorithm 1.

Initially, when \( k = 1 \), for each \( v_i \), we set \( F[v_i][v_{i+1}] = 1 \) and \( B[v_i][v_{i-1}] = \deg(v_i) \), and set all other entries of \( F[v_i] \) and \( B[v_i] \) to zero. In addition, we set \( L_i = 1, I_i = v_{i+1} \) and \( L'_i = 1, I'_i = v_{i-1} \). In the \( k \)-th iteration, with \( 1 < k \leq \lceil n/2 \rceil \), for each \( v_i \), we check whether \( v_i \) is visible to \( v_j \), with \( j = i + k \). If \( v_i \) is not visible to \( v_j \), then we do nothing. Otherwise, we set \( F[v_i][v_j] = L_i + 1 \) and increase \( L_i \) by one; similarly, we set \( B[v_j][v_i] = \deg(v_j) - L'_j \) and increase \( L'_j \) by one. Further, we set \( I_i = v_j \) and \( I'_j = v_i \).

It remains to show how to check whether \( v_i \) is visible to \( v_j \). By Lemma 2, we need to determine whether \( v' \) is a triangle witness vertex for \( v_i \) and \( v_j \). Since \( v' \) is the last visible vertex to \( v_i \) in the chain \( ch(v_{i+1}, v_{j-1}) \) in the CCW order, based on our definition, \( v' \) is the vertex \( I_i \). After knowing \( v' \), we then check whether \( v' \) is visible to \( v_j \), which can be done by checking whether \( B[v_j][v'] \) is zero, in constant time. If \( B[v_j][v'] \) is zero, then \( v' \) is not visible to \( v_j \) and \( v' \) is not a triangle witness vertex; otherwise, \( v' \) is visible to \( v_j \). (Note that we can also check whether \( F[v'][v_j] \) is zero.) In the following, we assume that \( v' \) is visible to \( v_j \). The next step is to determine whether
\(\angle_{v_i}(v', v_j) + \angle_{v_j}(v_i, v') + \angle_{v_i}(v_j, v_i) = \pi\). To this end, we must know the involved three angles. Similar to the discussion in Section 3, we have \(\angle_{v_j}(v_i, v_i) = \angle_{v_j}(F[v'[v_i], B[v'[v_i]])\), \(\angle_{v_i}(v', v_j) = \angle_{v_i}(F[v_i][v'], L_i + 1)\), and \(\angle_{v_j}(v_i, v') = \angle_{v_j}(\text{deg}(v_j) - L_j', B[v_j][v'])\). Thus, all these three angles can be obtained in constant time. Hence, the step of checking whether \(v_i\) is visible to \(v_j\) can be performed in constant time, which reduces another \(O(\log n)\) factor from the running time of the previous triangle witness algorithm in [6]. Algorithm 1 summarizes the whole algorithm. Clearly, the running time of our new algorithm is bounded by \(O(n^2)\).

**Theorem 1** Given the visibility angles and an ordered vertex sequence of a simple polygon \(P\), the improved triangle witness algorithm can reconstruct \(P\) (up to similarity) in \(O(n^2)\) time.

As discussed in [6], the above algorithm can also be used to determine whether the input angle data are consistent. Namely, if there is no polygon that can fit the input angle data, then the algorithm in Theorem 1 can be used to report this case as well.

**References**

[1] T. Asano, S.K. Ghosh, and T. Shermer. chapter 19: Visibility in the plane, in *Handbook of Computational Geometry*, J. Sack and J. Urrutia (eds.), pages 829–876. Elsevier, Amsterdam, The Netherlands, 2000.

[2] T. Biedl, S. Durocher, and J. Snoeyink. Reconstructing polygons from scanner data. In *Proc. of the 20th International Symposium on Algorithms and Computation*, volume 5878 of Lecture Notes in Computer Science, pages 862–871. Springer-Verlag, 2009.

[3] D. Bilò, Y. Disser, M. Mihalák, S. Suri, E. Vicari, and P. Widmayer. Reconstructing visibility graphs with simple robots. In *Proc. of the 16th International Colloquium on Structural Information and Communication Complexity*, volume 5869 of Lecture Notes in Computer Science, pages 87–99. Springer-Verlag, 2009.

[4] S.-H. Choi, S.Y. Shin, and K.-Y. Chwa. Characterizing and recognizing the visibility graph of a funnel-shaped polygon. *Algorithmica*, 14(1):27–51, 1995.

[5] P. Colley, A. Lubiw, and J. Spinrad. Visibility graphs of towers. *Computational Geometry: Theory and Applications*, 7(3):161–172, 1997.

[6] Y. Disser, M. Mihalák, and P. Widmayer. Reconstructing a simple polygon from its angles. In *Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139 of Lecture Notes in Computer Science, pages 13–24. Springer, 2010.

[7] G. Dudek, M. Jenkins, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

[8] H. Everett. *Visibility Graph Recognition*. PhD thesis, University of Toronto, Toronto, 1990.

[9] H. Everett and D. Corneil. Recognizing visibility graphs of spiral polygons. *Journal of Algorithms*, 11(1):1–26, 1990.

[10] H. Everett and D. Corneil. Negative results on characterizing visibility graphs. *Computational Geometry: Theory and Applications*, 5(2):51–63, 1995.
[11] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In Proc. of the 10th International Symposium on Algorithms and Computation, pages 93–102, 1999.

[12] L. Jackson and S. Wismath. Orthogonal polygon reconstruction from stabbing information. Computational Geometry: Theory and Applications, 23(1):69–83, 2002.

[13] J. O’Rourke and I. Streinu. The vertex-edge visibility graph of a polygon. Computational Geometry: Theory and Applications, 10(2):105–120, 1998.

[14] A. Sidlesky, G. Barequet, and C. Gotsman. Polygon reconstruction from line cross-sections. In Proc of the 18th Canadian Conference on Computational Geometry, pages 81–84, 2006.

[15] J. Snoeyink. Cross-ratios and angles determine a polygon. Discrete and Computational Geometry, 22:619–631, 1999.

[16] S. Suri, E. Vicari, and P. Widmayer. Simple robots with minimal sensing: From local visibility to global geometry. International Journal of Robotics Research, 27(9):1055–1067, 2008.

[17] S. Wismath. Point and line segment reconstruction from visibility information. International Journal of Computational Geometry and Applications, 10(2):189–200, 2000.
Appendix

**Algorithm 2: The Triangle Witness Algorithm [6]**

**Input:** \( n, (v_1, v_2, \ldots, v_n), \text{deg}(v_i) \), and the angles \( \angle_{v_i}(..., \cdot, \cdot, \cdot) \) for each \( 0 \leq i \leq n - 1 \).

**Output:** The edge set \( E \) of the visibility graph \( G \) for the target polygon \( P \).

/* All indices below are understood modulo \( n \). */

\( F \leftarrow \text{[array of } n \text{ empty maps]} \);
\( B \leftarrow \text{[array of } n \text{ empty maps]} \);
\( E \leftarrow \emptyset \);

for \( i \leftarrow 0 \) to \( n - 1 \) do

\( E \leftarrow E \cup \{e(v_i, v_{i+1})\} \);
\( F[v_i][v_{i+1}] \leftarrow 1 \);
\( B[v_i][v_{i-1}] \leftarrow \text{deg}(v_i) \);

for \( k \leftarrow 2 \) to \( \lceil n/2 \rceil \) do

for \( i \leftarrow 0 \) to \( n - 1 \) do

\( j \leftarrow i + k \);

for \( l \leftarrow i + 1 \) to \( j - 1 \) do

if \( v_l \in F[v_i] \) and \( v_l \in B[v_j] \) then

\( \angle_{v_i}(v_l, v_j) \leftarrow \angle_{v_i}(F[v_i][v_l], |F[v_i]| + 1) \);
\( \angle_{v_j}(v_l, v_i) \leftarrow \angle_{v_j}(\text{deg}(v_j) - |B[v_j]|, B[v_j][v_l]) \);
\( \angle_{v_l}(v_j, v_i) \leftarrow \angle_{v_l}(F[v_l][v_j], B[v_l][v_i]) \);

if \( \angle_{v_i}(v_l, v_j) + \angle_{v_j}(v_i, v_l) + \angle_{v_l}(v_j, v_i) = \pi \) then

\( E \leftarrow E \cup \{e(v_i, v_j)\} \);
\( F[v_i][v_j] \leftarrow |F[v_i]| + 1 \);
\( B[v_j][v_i] \leftarrow \text{deg}(v_j) - |B[v_j]| \);

abort the innermost loop;