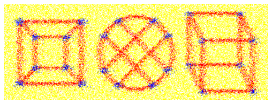

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 0, no. 0, pp. 0–0 (0)
DOI:

Reconstructing Generalized Staircase Polygons with Uniform Step Length

Nodari Sitchinava[1] *Darren Strash*[2]

[1]Department of Information and Computer Sciences, University of Hawaii at Manoa, USA.
[2]Department of Computer Science, Hamilton College, Clinton, NY, USA.

Abstract

Visibility graph reconstruction, which asks us to construct a polygon that has a given visibility graph, is a fundamental problem with unknown complexity (although visibility graph recognition is known to be in PSPACE). As far as we are aware, the only class of *orthogonal* polygons that are known to have efficient reconstruction algorithms is the class of orthogonal convex fans (*staircase* polygons) with uniform step lengths. We show that two classes of uniform step length polygons can be reconstructed efficiently by finding and removing rectangles formed between consecutive convex boundary vertices called tabs. In particular, we give an $O(n^2m)$-time reconstruction algorithm for orthogonally convex polygons, where n and m are the number of vertices and edges in the visibility graph, respectively. We further show that reconstructing a monotone chain of staircases (a histogram) is fixed-parameter tractable, when parameterized on the number of tabs, and polynomially solvable in time $O(n^2m)$ under alignment restrictions. As a consequence of our reconstruction techniques, we also get recognition algorithms for visibility graphs of these classes of polygons with the same running times.

Submitted: Reviewed: Revised: Accepted: Final:

Published:

Article type: Communicated by:

This material is based upon work supported by the National Science Foundation under Grant No. 1533823.

E-mail addresses: nodari@hawaii.edu (Nodari Sitchinava) dstrash@hamilton.edu (Darren Strash)

1 Introduction

Visibility graphs, used to capture visibility in or between polygons, are simple but powerful tools in computational geometry. They are integral to solving many fundamental problems, such as routing in polygons, and art gallery and watchman problems, to name a few. Efficient, and even worst-case optimal, algorithms exist for computing a visibility graph from an input polygon [17]; however, comparatively little is known about the reverse direction: the so-called visibility graph *recognition* and *reconstruction* problems.

In this paper, we study *vertex-vertex visibility graphs*, which are formed by visibility between pairs of vertices of a polygon. Given a graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, the visibility graph recognition problem asks if G is the visibility graph of *some* polygon. Similarly, the visibility graph reconstruction problem asks us to construct a polygon with G as a visibility graph. Surprisingly, recognition of simple polygons is only known to be in PSPACE [14], and it is still unknown if simple polygons can be reconstructed in polynomial time. Therefore, current solutions are typically for restricted classes of polygons. Even characterizations are only known for special classes of visibility graphs [18, 21], and only four necessary conditions are known [16] for standard visibility graphs.

1.1 Special Classes of Polygons

A well-known result due to ElGindy [12] is that every maximal outerplanar graph is a visibility graph and a polygon can be reconstructed from every such graph in polynomial time. Other special classes rely on a unique configuration of reflex and convex chains, which restrict visibility.

For instance, spiral polygons [15] and tower polygons [8] (also called funnel polygons), can be reconstructed in linear time, and each consists of one and two reflex chains, respectively. Further, 2-spirals can be reconstructed in polynomial time [2], as can a more general class of visibility graphs related to 3-matroids [3]. Finally convex polygons with a single convex hole [7] can be reconstructed in polynomial time.

For monotone polygons, Colley [9, 10] showed that if each face of a maximal outerplanar graph is replaced by a clique on the same number of vertices, then the resulting graph is a visibility graph of some uni-monotone polygon (monotone with respect to a single edge), and such a polygon can be reconstructed if the Hamiltonian cycle of the boundary edges is known. However, not every uni-monotone polygon (even those with uniformly spaced vertices) has such a visibility graph [13]. Finally, Evans and Saeedi [13] characterized terrain visibility graphs, which consist of a single monotone polygonal line.

Orthogonal polygons. Surprisingly little is known about visibility graphs of orthogonal polygons: we are only aware of results for *orthogonal convex fans*, which consist of a single staircase and an extra vertex. These are also known as *staircase polygons*. Abello and Eğecioğlu [1] show that orthogonal convex fans

with uniform side-length can be reconstructed in linear time; however, their construction relies on a simpler definition of visibility, which allows for blocking vertices. Furthermore, Abello et al. [4] show that general orthogonal convex fans are *recognizable* in polynomial time, but reconstruction is still open even for this simple class of orthogonal polygons.

Other algorithms for orthogonal polygons use different visibility representations, which seem to be easier to work with. For example, researchers have looked into edge-edge visibility and vertex-edge visibility. For orthogonal polygons, edge-edge visibility graphs [20, Section 7.3] have been shown to consist of two disjoint trees and the polygon can be reconstructed when these trees "mesh well". For vertex-edge visibility, orthogonal polygons can be reconstructed entirely from "stabs" [19]—for each vertex, we are given the edges hit by horizontal (or vertical) rays shot from the vertex—in $O(n \log n)$ time. However, we also must be given the Hamiltonian cycle of the boundary edges with the input. See Asano et al. [5] or Ghosh [16] for a thorough review of results on visibility graphs.

1.2 Our Results

In this work, we investigate reconstructing polygons consisting of multiple uniform step length staircases. To the best of our knowledge, the *only* pure visibility reconstruction result for orthogonal polygons is for a single uniform step length staircase [1]. We first show that orthogonally convex polygons can be reconstructed in $O(n^2m)$ time. We further show that reconstructing orthogonal uni-monotone polygons (also known as *histograms*) is fixed-parameter tractable, when parameterized on the number of horizontal boundary edges that are incident to two convex vertices in the polygon. We also provide an $O(n^2m)$ time algorithm under alignment restrictions. As a consequence of our reconstruction technique, we can also recognize the visibility graphs of these classes of polygons with the same running times.

2 Preliminaries

Let P be a polygon on n vertices. We say that a point p *sees* a point q (or p and q are *visible*) in polygon P if the line segment $\overline{pq}$ does not intersect the exterior of P. Under this definition, visibility is allowed along edges and through vertices. If we restrict our attention to visibility between vertices of the polygon, we get a natural pairwise relationship between vertices, giving us a *visibility graph*.

Definition 1 (visibility graph) *A visibility graph $G_P = (V_P, E_P)$ of polygon P has a vertex $v_p \in V_P$ for each vertex p of P, and an edge $(v_p, v_q) \in E_P$ if and only if vertices p and q are visible in P.*

Due to the one-to-one relationship between polygon and graph vertices, we reference the vertex of a visibility graph as we would a vertex of the polygon.

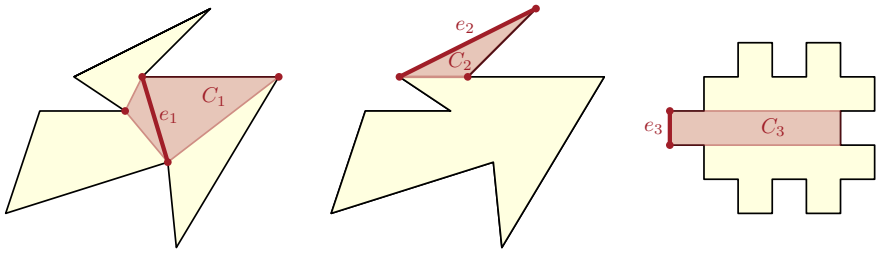


Figure 1: Maximal convex regions on vertices of polygons (e.g., C_1, C_2, and C_3) are maximal cliques in visibility graphs, and 1-simplicial edges (e.g., e_1, e_2, and e_3) are each in exactly one maximal clique.

For our visibility graph discussion, we adopt standard notation for graphs and polygons. In particular, for a graph $G = (V, E)$, we denote the *neighborhood* of a vertex $v \in V$ by $N(v) = \{u \mid (v, u) \in E\}$, the degree of v by $deg(v) = |N(v)|$, and the number of vertices and edges by $n = |V|$ and $m = |E|$, respectively. For a visibility graph $G_P = (V_P, E_P)$ of a polygon P, we call an edge in G_P that is an edge of P a *boundary edge*. Other edges (diagonals in P) are *non-boundary edges*.

Finally, critical to our proofs is the fact that a maximal clique in G_P corresponds to a maximal (in the number of vertices) convex region $R \subseteq P$ whose vertices are defined by vertices of P. We say that an edge (u, v) is 1-*simplicial* if the common neighborhood $N(u) \cap N(v)$ is a clique, or equivalently (u, v) is in exactly one maximal clique[1]. The intuition behind why we consider 1-simplicial edges is that, in orthogonal polygons with edges of uniform length, boundary edges between convex vertices are 1-simplicial, with the vertices of the clique forming a rectangle. (See Figure 1.) Note, however, that non-boundary edges may also be 1-simplicial.

For our running times, we rely on the following observation.

Observation 1 *We can test if (u, v) is 1-simplicial and in a maximal k-clique in time $O(kn)$ with an adjacency list data structure, or time $O(k^2 + n)$ with an adjacency matrix.*

Proof: Compute $X = N(u) \cap N(v)$ in $O(n)$ time by marking vertices in $N(u)$ and checking for marked vertices in $N(v)$, then check that $|X| = k$ and X is a clique. This last step takes $O(kn)$ time with an adjacency list, or $O(k^2)$ with an adjacency matrix. □

[1]This is not to be confused with *simplicial edges*, which are defined elsewhere to be edges (u, v) such that for every $w \in N(u)$ and $x \in N(v)$, w and x are adjacent. Instead, this definition parallels the usage for vertices: A vertex v is *simplicial* if $N(v)$ forms a clique, or equivalently v is in exactly one maximal clique.

We note that if $k = O(n)$ this test takes time $O(n^2)$ and, if $k = O(1)$, the test takes time $O(n)$.

3 Uniform-Length Orthogonally Convex Polygons

We first turn our attention to a restricted class of orthogonal polygons that have only uniform-length (or equivalently, unit-length) edges. Let P be an orthogonal polygon with uniform-length edges such that no three consecutive vertices on P's boundary are collinear, and further let P be *orthogonally convex*[2]. We call P a *uniform-length orthogonally convex polygon* (UP). Note that every vertex v_i on P's boundary is either convex or reflex. We call boundary edges between two convex vertices in a uniform-length orthogonal polygon P *tabs* and the endvertices of a tab *tab vertices*. We reconstruct the polygon by computing the clockwise ordering of vertices of the UP.

Note that the boundary of a UP consists of four tabs connected via staircases. For ease of exposition, we assume that the UP is embedded in $\mathbb{R}^2$ with polygon edges axis-aligned. We call the tab with the largest y-coordinate the north tab, and we analogously name the others the south, east, and west tabs. Following this convention, we refer to the four boundary staircases as northwest, northeast, southeast, and southwest. Furthermore, note that it is possible for two (opposite) staircases to have more vertices than the other two (also opposite) staircases. We call the staircases with more vertices *long staircases* and others *short*.

We only consider polygons with more than 12 vertices, which eliminates many special cases. Smaller polygons can be solved in constant time via brute force.

We first introduce several structural lemmas which help us to identify all convex vertices in a UP, which is key to our reconstruction.

Lemma 1 *For every convex vertex u in a UP, there is a convex vertex v, such that $(u, v) \in E_P$ and (u, v) is 1-simplicial.*

Proof: If u is a tab vertex, then the tab vertex v that is u's boundary neighbor is convex and (u, v) is 1-simplicial. Otherwise, without loss of generality, suppose that u is on the northwest staircase. Then there is a convex vertex v on the southeast staircase that is visible from u. Edge (u, v) is in exactly one maximal clique, consisting of u, v, the reflex vertices within the rectangle R defined by u and v as the opposite corners, and any other corners of R that are convex vertices of the polygon. □

Furthermore, we have the following lemma, which allows us to differentiate between convex and reflex vertices.

[2]That is, any two points in P can be connected by a staircase contained in P.

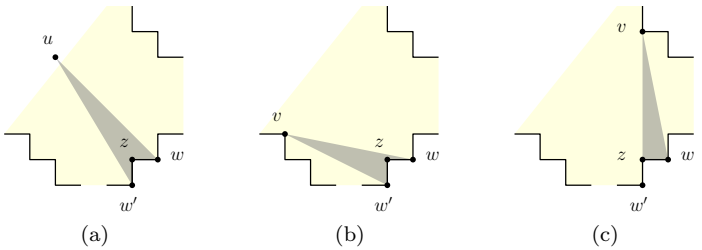


Figure 2: Illustration of the proofs of (a) Observation 2 and (b)-(c) Observation 3. The vertex u on the northwest staircase can be reflex or convex.

Lemma 2 *For every visible pair of vertices u and v in a UP, if u or v is a reflex vertex, then the edge $(u, v) \in E_P$ is not 1-simplicial.*

Combining Lemmas 1 and 2 gives us a simple method to determine convex and reflex vertices: test if each edge is in more than one maximal clique. Mark all vertices incident to 1-simplicial edges as convex, and mark the rest as reflex.

We prove Lemma 2 below, but first we make some necessary observations.

Observation 2 *If a vertex u on the northwest staircase of a UP sees a reflex vertex z on the southeast staircase and the slope of the line supporting the line segment $\overline{uz}$ is non-positive, then u sees both (convex) boundary neighbors w and w' of z.*

Proof: We will prove that w, the horizontal boundary neighbor of z, is visible from u. The proof for visibility of w', the vertical boundary neighbor of z, is symmetric. See Figure 2(a) for an illustration.

First, observe that in a UP, no boundary edge on the northeast and the southwest staircases blocks visibility between the vertices of the northwest staircase and the vertices of the southeast staircase, and, consequently, between u and w.

Next, by viewing u as the origin of the Cartesian coordinate system, z lies in the southeast quadrant (inclusive of the axis), since the slope of the line supporting $\overline{uz}$ is non-positive. And since $\overline{zw}$ is a horizontal edge, w also lies in the southeast quadrant. Therefore, no boundary edge of the northwest staircase blocks the visibility between u and w (recall that our definition of visibility allows visibility along the polygon edges).

Finally, by viewing w as the origin, we can similarly see that no edge of the southeast staircase blocks the visibility between u and w. □

Observation 3 *Let a reflex vertex v be on the southwest or northeast staircase of a UP and a reflex vertex z be on the southeast staircase. If the slope of the line supporting the line segment $\overline{vz}$ is non-positive, then v sees both (convex) boundary neighbors w and w' of z.*

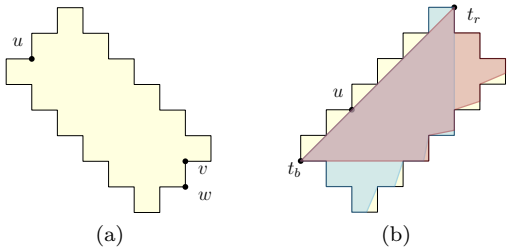


Figure 3: Illustration of the proof that an edge (u, v) is not 1-simplicial when both u and v are reflex. The proof demonstrates the existence of a convex vertex w visible from both u and v.

Proof: Consider the case when v is on the southwest staircase (the proof of the case when v is on the northeast staircase is symmetric). See Figure 2(b)-(c) for an illustration.

First, observe that no boundary edge on the northwest and northeast staircase blocks the visibility between the vertices of the southwest staircase and the vertices of the southeast staircase, and, consequently, between v and w.

Next, observe that the slope of the line supporting the edges from *any* reflex vertex on the southwest staircase and *any* vertex on the southeast staircase is at least $-\tan(\pi/4)$ and, therefore, no vertex on the southwest staircase can block the visibility between them.

Finally, by viewing w (resp., w') as the origin of the Cartesian coordinate system, we can see that v is in the northwest quadrant (inclusive of axis) and, therefore, no boundary edge on the southeast staircase can block the visibility between w (resp., w') and v. □

We are ready to prove Lemma 2.

Proof: Let $(u, v) \in E_P$ and suppose that at least one of u and v is reflex.

Case 1: Both u and v are reflex. Then they belong to a maximal clique consisting of all reflex vertices and no convex vertices. We will show that both u and v also see some convex vertex w, therefore, u, v and w are part of another maximal clique. For concreteness of exposition, we orient the polygon so u is always on the northwest staircase. Further, if all staircases have the same number of vertices, we arbitrarily choose the northwest and southeast staircases to be called short, and the others long. There are two cases to consider:

(a) **Both u and v are on short staircases.** (See Figure 3(a)). Observe, that every reflex vertex on a short staircase sees all vertices on the other short staircase. Then, if u and v are on the same (northwest) staircase, there is a convex vertex w on the southeast staircase, visible to both u and v. If v is on the southeast staircase, w is either of the two convex boundary neighbors of v. Since w is a boundary neighbor of v, it is visible from v.

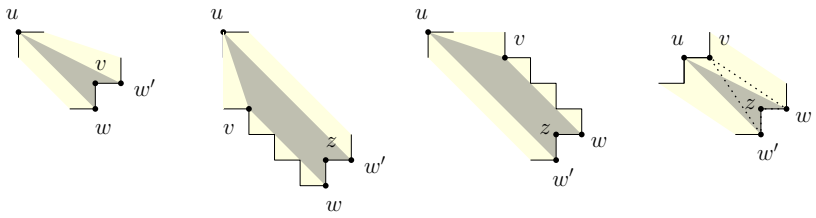


Figure 4: Illustration of the proof that an edge (u, v) is not 1-simplicial when u is convex and v is reflex. For each of the four cases of when v is on the four different staircases, there is a pair of convex vertices w and w' on the southeast staircase, which are visible to both u and v, but not to each other.

(b) **At least one of u and v is on a long staircase.** (See Figure 3(b)). Vertex u sees the right tab vertex t_r of the north tab and the bottom tab vertex t_b of the west tab. Since the northwest staircase is long, the union of reflex vertices visible from t_r and t_b is the set of all reflex vertices in the polygon. Therefore, every reflex vertex v must see at least one of t_r or t_b.

Case 2: One of the vertices, u or v, is convex. Without loss of generality, let it be u (again, on the northwest staircase), and let v be reflex. Let P_u be the polygon consisting of u and all reflex vertices visible from u. For every reflex vertex v in P_u, we will identify two convex vertices w and w' on the southeast staircase, which are visible to both u and v. Since w and w' are convex vertices on the same staircase, they are invisible to each other and, therefore, u, v, w and u, v, w' will be part of two distinct maximal cliques.

Let us consider the four possible locations for v (see Figure 4):

(a) v **is on the southeast staircase.** Let w and w' be the two convex boundary neighbors of v. Clearly, v sees w and w'. Moreover, since u sees v and v is on the opposite staircase from u, the slope of the line supporting the segment $\overline{uv}$ is non-positive. By Observation 2, u must also see w and w'.

(b) v **is on the southwest staircase.** Let z be the lowest reflex vertex on the southeast staircase (adjacent to the south tab's right vertex) and let w and w' be the two convex boundary neighbors of z. Vertex z is visible from v because both v and z are reflex. Since z is the lowest reflex vertex, the slope of the line supporting the segment $\overline{vz}$ is non-positive (the slope is 0 if v is the lowest reflex vertex on the southwest staircase). Thus, by Observation 3, w and w' are visible from v. Moreover, since u sees v and v is on the southwest staircase, z must be visible from u with even smaller slope of the line supporting segment $\overline{uz}$, i.e., the slope is also non-positive. Therefore, by Observation 2, w and w' are also visible from u.

(c) v **is on the northeast staircase.** Let z be the highest reflex vertex on the southeast staircase (adjacent to the east tab's bottom vertex) and let w

and w' be the two convex boundary neighbors of z. Again, vertex z is visible from v because both v and z are reflex. We can see that the slope of the line supporting the segment $\overline{vz}$ is negative by viewing v as the origin of the Cartesian coordinate system and observing that z is in the southeast quadrant. Thus, by Observation 3, w and w' are visible from v. At the same time, z is visible from u because v is visible from u and z is below and to the right of (or directly below) v. Moreover, the slope of the line supporting segment uz is also non-positive. Therefore, by Observation 2, w and w' are also visible from u.

(d) v **is on the northwest staircase.** Since v is visible from u, v must be a reflex boundary neighbor of u. Observe, that because P_u is bounded from all sides, it must contains a non-empty subset Z of reflex vertices from the southeast staircase. Moreover, there exists a vertex $z \in Z$, which is not axis-aligned with u, i.e., the slope of the line supporting the segment $\overline{uz}$ is strictly negative. Let w and w' be the two convex boundary neighbors of z. By definition of P_u, z is visible from u, therefore, by Observation 2, w and w' are visible from u. Again, z is visible from v because both v and z are reflex. Since the slope of the line supporting segment $\overline{uz}$ is strictly negative and the polygon has uniform length boundaries, the slope of the line supporting segment $\overline{vz}$ must be non-positive. Therefore, by Observation 2, w and w' are also visible from v.

Therefore, (u, v) is not 1-simplicial. □

Lemma 2 states that only edges between convex vertices can be 1-simplicial. Combining this lemma with Lemma 1, we can identify all convex vertices by checking for each edge (u, v) if $N(u) \cap N(v)$ is a clique in $O(n^2)$ time, leading to the following lemma.

Lemma 3 *We can identify all convex and reflex vertices in a visibility graph of a UP in $O(n^2 m)$ time.*

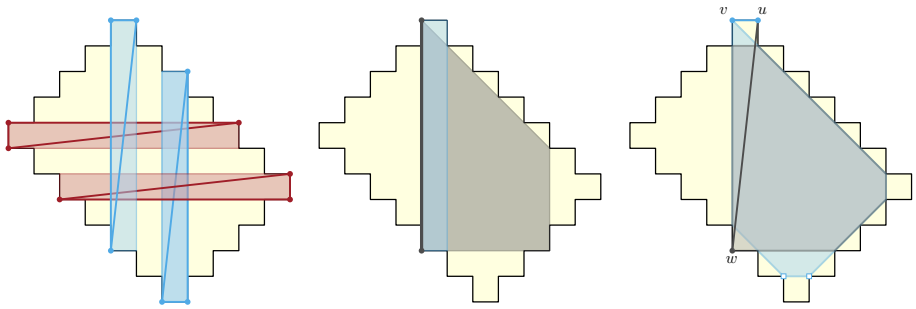
We now divide the class of uniform-length orthogonal polygons into two classes of polygons, which we call *regular* and *irregular*.

Definition 2 (regularity) *We call a UP* regular *if each of its staircase boundaries have the same number of vertices. Otherwise, we call it* irregular*, consisting of two* long *and two* short *staircases.*

We restrict our attention to irregular uniform-length orthogonally convex polygons (IUPs); however, similar methods work for their regular counterparts.

3.1 Irregular Uniform-Length Orthogonally Convex Polygons

Let G_P be the visibility graph of IUP P. Our reconstruction algorithm first computes the four tabs, then assigns the convex and reflex vertices to each



(a) (b) (c)

Figure 5: Steps to identify the tabs. (a) Each tab clique (highlighted) has three edges between convex vertices. (b) Vertical (also horizontal) non-boundary edges are not 1-simplicial. (c) After removing these edges, the middle vertex u on each remaining 2-path is a tab vertex, and in this instance the other tab vertex v sees more reflex vertices than w.

staircase. The following structural lemma helps us to find the tabs. We assume that we have already computed the convex and reflex vertices in $O(n^2m)$ time.

Lemma 4 *In every IUP there are exactly four 7-vertex maximal cliques that individually contain exactly three convex vertices. Each such clique contains exactly one tab, and each tab is contained in exactly one of these cliques.*

Proof: First note that each of the four tabs is in exactly one such maximal 7-clique. Further, any other clique that contains three convex vertices has at least nine vertices. Each convex vertex must be on a different staircase with its two boundary neighbors, which are therefore distinct. □

We note that it is not necessary to identify the four tabs explicitly to continue with the reconstruction. There are only $7^4 = O(1)$ choices of tabs (one from each 7-clique of Lemma 4), thus we can try all possible tab assignments, continue with the reconstruction and verify that our reconstruction produces a valid IUP P with the same visibility graph. Since there are only $O(1)$ choices for tabs, this would add no additional running time to our algorithm.

However, we show how to explicitly find the four tabs in the proof of the following lemma.

Lemma 5 *We can identify the four tabs of an IUP in $O(nm)$ time.*

Proof: We compute the four 7-vertex maximal cliques of Lemma 4 in $O(nm)$ time using Observation 1. These cliques have exactly three convex vertices each, and tabs are incident to two convex vertices, narrowing our choice of tab down to $4 \cdot \binom{3}{2} = 12$ edges. See Figure 5(a). Four of these are vertical or horizontal

(non-boundary) edges, which we can detect and eliminate, as they are not 1-simplicial: these edges are in both a clique with both tab vertices, and a clique containing a tab vertex, another convex vertex, and reflex vertices not in the tab clique. See Figure 5(b). We have eight remaining edges to consider. These eight edges form four disjoint paths on two edges, and the middle vertex on each path is a tab vertex.

Note that these "middle" tab vertices are on the long staircases. Let one of them be called u. Now it remains to find u's neighbor on its tab. Vertex u has two candidate neighbors; let's call them v and w. Just for concreteness, let's say u is the vertex of the north tab on the northeast staircase. See Figure 5(c).

Suppose, without loss of generality, that v has more reflex neighbors than w, then v is u's neighbor on a tab, because it sees reflex vertices on the whole northeast and southeast staircases, while w sees only a subset of those. Otherwise v and w have the same number of reflex neighbors, which only happens when w sees every reflex vertex on the southeast and northeast staircases. Then either v or w has more convex neighbors. Suppose, without loss of generality, that v is a tab vertex, then v has fewer convex neighbors than w. To see why, note that since u is on the northeast (long) staircase, v is on the northwest (short) staircase. Vertex v has convex neighbors u, w, and every convex vertex on the southeast (short) staircase. Likewise, w has convex neighbors v, u, every convex vertex on the northeast (long) staircase (including u) and one vertex on the southeast (short) staircase.

We can perform these checks for all such pairs v and w, thereby computing all tabs. Note that, if we have already distinguished convex and reflex vertices, this step takes time $O(nm)$: $O(nm)$ time to compute the four cliques containing tabs, and $O(n)$ to count the number of reflex and convex vertices visible from each tab vertex candidate. □

We pick one tab arbitrarily to be the north tab. We conceptually orient the polygon so that the northwest staircase is short and the northeast staircase is long. We do this by computing *elementary cliques*, which identify the convex vertices on the short staircase.

Definition 3 (elementary clique) *An* elementary clique *in an IUP is a maximal clique that contains exactly three convex vertices: one from a short staircase, and one from each of the long staircases. (See Figure 6(a).)*

Lemma 6 *We can identify the elementary cliques containing vertices on the northwest staircase in $O(nm)$ time.*

Proof: Each elementary clique C contains a 1-simplicial edge, as C is maximal and two convex vertices in C must be on opposite staircases. Using Observation 1, we compute all 1-simplicial edges in maximal cliques on seven or nine vertices in $O(nm)$ time, and keep the cliques that are elementary cliques—those that contain three convex vertices (and either four or six reflex vertices).

Let k_{NW} be the number of convex vertices on the northwest staircase. We number these convex vertices from v_0 to $v_{k_{NW}-1}$ in order along the northwest

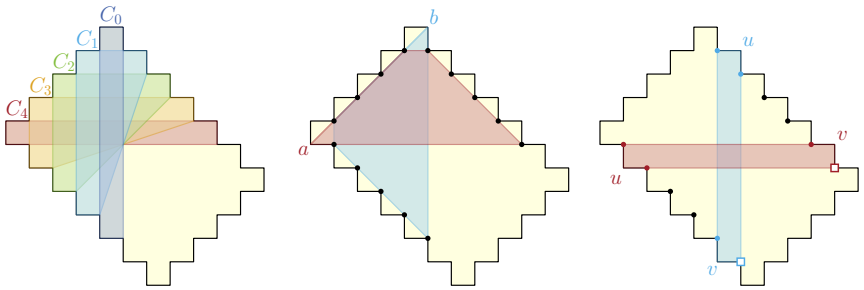


Figure 6: Elements of our reconstruction. (a) The elementary cliques $C_0, \ldots, C_4$ interlock along a short staircase. (b) The tab vertices a and b see unique reflex vertices on long staircases. (c) Unassigned reflex vertices (squares) are assigned by forming rectangles between the convex vertices u and v and their already-assigned reflex boundary neighbors (circles).

staircase from the north tab to the west tab. Denote by C_i the unique elementary clique containing v_i. Then C_0 is the unique maximal (elementary) clique containing the north tab. Furthermore, each clique C_i contains a set of reflex vertices R_i such that $|R_i \cap C_{i+1}| = 3$, and for $j \notin \{i-1, i, i+1\}$, $R_i \cap C_j = \emptyset$. Therefore, given the elementary cliques $C_0, \ldots, C_i$, we can compute the elementary clique C_{i+1} by searching for the remaining elementary clique containing three reflex vertices in R_i. (Note that the elementary clique C_{i-1} also contains vertices in R_i, but C_{i-1} has already been discovered.) Once we reach an elementary clique containing a tab, then we have computed all elementary cliques on the northwest staircase. This tab is the west tab and we are finished. □

Note that, if our sole purpose is to reconstruct the IUP P, we have sufficient information. The number of elementary cliques gives us the number of vertices on a short staircase of P, from which we can build P. However, in what follows, we can actually map all vertices to their positions in the IUP, a technique which we later use to build a recognition algorithm for IUPs.

First, we show how to assign all convex vertices from the elementary cliques to each of the three staircases, using visibility of the north and west tab vertices. Note, constructing the elementary cliques with Lemma 6 also gives us the west tab, since it is contained in the last elementary clique on the northwest staircase.

Lemma 7 *We can identify the convex vertices on the northwest staircase in $O(n)$ time.*

Proof: The northwest staircase contains the convex vertices of the elementary cliques from Lemma 6 that cannot be seen by any of the north or west tab vertices. The staircase further contains the left vertex of the north tab and the

top vertex of the west tab (which can be identified by the fact that they are tab vertices that do not see either vertex of the other tab). □

We can repeat the above process to identify the convex vertices of the southeast staircase. However, we might not yet be able to identify tabs as south or east. Thus, we will obtain two possible orderings of the convex vertices on the southeast staircase. Next, we show how to assign convex vertices to the long staircases. In the process we determine south and east tabs, and consequently, identify the correct ordering of convex vertices on the southeast staircase.

Lemma 8 *We can assign the remaining convex vertices in $O(n^2)$ time.*

Proof: Let $\mathcal{W}$ and $\mathcal{E}$ be the sets of all convex vertices on the southwest and northeast staircases (which we are computing) and let $\mathcal{W}_0$ and $\mathcal{E}_0$ be the sets of convex vertices on the southwest and northeast staircases that are already known from the elementary cliques from Lemma 7. Note that the order of these vertices has not been determined.

Let $N_c(v)$ denote the set of convex neighbors on the opposite staircase of a vertex v. We now illustrate how to discover vertices in $\mathcal{W}$ and $\mathcal{E}$. Suppose for the moment that we can select the convex vertex $w_0 \in \mathcal{W}_0$ that is the northernmost vertex in $\mathcal{W}_0$. There is exactly one convex neighbor of w_0 that has not yet been discovered, and it is in $N_c(w_0) \subseteq \mathcal{E}$. Call this convex vertex e. We note that a maximal clique (a rectangle) is formed between vertices w_0, e, and their reflex boundary neighbors, and therefore we have discovered a new vertex in $\mathcal{E}$ by computing a rectangle spanning the polygon. See Figure 6(c). However, note that we do not need to begin with the northernmost (convex) vertex in $\mathcal{W}_0$; we can discover vertices in $\mathcal{E}$ with any vertex $w_0 \in \mathcal{W}_0$. All of w_0's undiscovered convex neighbors are in $\mathcal{E}$, and at least one vertex in $\mathcal{W}_0$ (e.g., the northernmost) has undiscovered neighbors in $\mathcal{E}$. Similarly, with vertices in $\mathcal{E}_0$ we will discover at least one new $w \in \mathcal{W}$. We can then iteratively build sets $\mathcal{W}$ and $\mathcal{E}$ by computing the sets $\mathcal{E}_i = (\cup_{w \in \mathcal{W}_{i-1}} N_c(w)) \setminus \mathcal{E}_{i-1}$ and $\mathcal{W}_i = (\cup_{e \in \mathcal{E}_{i-1}} N_c(e)) \setminus \mathcal{W}_{i-1}$. We then identify all vertices of the southwest and northeast staircases as $\mathcal{W} = \cup_i \mathcal{W}_i$ and $\mathcal{E} = \cup_i \mathcal{E}_i$.

To order the vertices along the southwest staircase, note that the sets $\mathcal{W}_i$ should appear in order of increasing i from top to bottom. Also note that if one were to assign the vertices of $\mathcal{W}_i$ to a staircase from top to bottom, each vertex w_i in this order would see fewer vertices of $\mathcal{E}_{i-1}$. Thus, we can order the vertices within each $\mathcal{W}_i$. The argument for ordering vertices of $\mathcal{E}_i$ is symmetric.

Computing the set $\mathcal{E}_i$ takes time $O(|\mathcal{W}_{i-1}| \cdot n)$: we evaluate all $O(n)$ neighbors of each neighbor in $\mathcal{W}_{i-1}$. Likewise, computing $\mathcal{W}_i$ takes time $O(|\mathcal{E}_{i-1}| \cdot n)$. Thus, overall it takes $O((|\mathcal{W}| + |\mathcal{E}|) \cdot n) = O(n^2)$ time to construct (and order) sets $\mathcal{W}$ and $\mathcal{E}$. □

We can now choose the south and east tabs: a vertex on the east (south) tab can see convex vertices on the southwest (northeast) staircase.

Now all convex vertices are assigned to staircases, and all that remains is to assign any remaining reflex vertices.

Lemma 9 *We can assign the reflex vertices to each staircase in $O(n^2)$ time.*

Proof: Now that the convex vertices are also ordered on the long staircases, we can assign the reflex vertices to each staircase. We first assign those reflex vertices that are seen from the tab vertices. Let a and b be vertices on different tabs such that a sees b and $\overline{ab}$ contains all reflex vertices of a short staircase (see Figure 6(b)). For ease of discussion, we assume that a and b are on the west and north tabs, respectively. Let $\mathcal{R}$ be the set of all reflex vertices of the IUP. The set $\mathcal{R}_0 = N(a) \cap N(b) \cap \mathcal{R}$ contains all reflex vertices from the northwest staircase, plus two extra reflex vertices: the boundary neighbors of a and b, each on a different long staircase. The remaining vertices $N(a) \setminus \mathcal{R}_0$ are on the northeast (long) staircase and $N(b) \setminus R_0$ are on the southwest (long) staircase. Let k be the number of reflex vertices on the northwest staircase, then $|\mathcal{R}_0| = k + 2$ and $|N(b) \setminus R_0| = |N(a) \setminus \mathcal{R}_0| = k$.

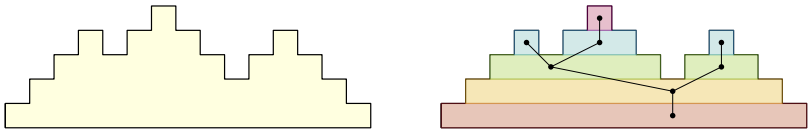
Let ℓ be the number of reflex vertices on the southwest (or equivalently, northeast) staircase, and let $w_1, w_2, \ldots, w_\ell$, and $e_1, e_2, \ldots, e_\ell$ be the reflex vertices on the southwest and northeast staircases respectively (which we are computing), where subscripts indicate north to south order along each staircase. In contrast to w_1 and e_1, we have already discovered the vertices $w_2, \ldots, w_{k+1}$ and $e_2, \ldots, e_{k+1}$ – though we do not yet know their order. However, the reflex vertices on a single staircase can easily be ordered in time $O(n^2)$ time by intersecting the neighborhoods of the (already-ordered) consecutive convex vertices on the staircase.

We first compute the reflex boundary neighbors of a and b, which will be w_1 and e_1, respectively. Note that, w_1 is contained in a maximal 4-clique that is a degenerate rectangle (a horizontal line segment) containing a, w_1, e_{k+1}, and e_{k+1}'s convex boundary neighbor to the east, x, which is the $(k+1)$-th convex vertex on the northeast staircase. The vertex w_1 can be found in time $O(n)$ by computing this 4-clique from the 1-simplicial edge (a, x). The vertex e_1 can be assigned analogously. Note also that this gives us all reflex vertices on the northwest staircase, which consists of the set of vertices $\mathcal{R}_0 \setminus \{w_1, e_1\}$.

To assign the remaining reflex vertices to their staircases, we will conceptually build axis-aligned rectangles that span from one long staircase to the other, similar to the proof of Lemma 8. These rectangles are maximal cliques consisting of two convex vertices u and v on different long staircases, two known reflex boundary neighbors of u, and one known reflex boundary neighbor of v (see Figure 6(c)). These rectangles contain one yet-unassigned reflex vertex, which is boundary neighbor of v, which is then assigned to v's staircase.

Let u be the (convex) boundary neighbor of both w_1 and w_2, and let v be the boundary neighbor of e_{k+1} that is not a neighbor of e_k. Then (u, v) is a 1-simplicial edge in a maximal 6-clique (rectangle) with a single unassigned reflex vertex, which is e_{k+2}. A symmetric argument can be used to discover w_{k+2} from e_1, e_2 and w_{k+1}. We alternate discovering vertices from the southwest and northeast staircases until we have assigned all reflex vertices.

Note that since convex vertices are known and in order along their staircases, the convex vertex v can be be computed in $O(1)$ time from u: if u is the i-th



(a) (b)

Figure 7: (a) A histogram with three tabs and (b) its decomposition into touching rectangles with a contact graph that is a tree.

convex vertex on a long staircase, then v is the $(i+k)$-th vertex on the opposite long staircase. Given the two convex vertices u and v, it takes $O(n)$ time to compute the maximal 6-clique of the rectangle containing them, and $O(1)$ time to determine the new reflex vertex. We must do this for $O(n)$ rectangles. Hence, the total time to discover these new reflex vertices is $O(n^2)$. □

Observe that within each staircase, boundary edges are formed only between convex vertices and their reflex neighbors. Thus, we can order reflex vertices on each staircase by iterating over the staircase's convex vertices (order of which is determined in Lemmas 6-8) and we are done. The running time of our algorithm is dominated by the time to differentiate convex and reflex vertices, giving us the following result:

Theorem 1 *In $O(n^2m)$ time, we can reconstruct an IUP from its visibility graph.*

4 Uniform-Length Histogram Polygons

In this section we show how to reconstruct a more general class of polygons: those that consist of a chain of alternating up- and down-staircases with uniform step length, which are monotone with respect to a single (longer) *base edge*. Such polygons are uniform-length *histogram polygons* [11], but we simply call them *histograms* for brevity (see Figure 7(a) for an example). We refer to the two convex vertices comprising the base edge as *base vertices*. Furthermore, we refer to top horizontal boundary edges incident to two convex vertices as *tab edges* or just *tabs* and their incident vertices as *tab vertices*.

We briefly note that, for histograms with two staircases, reconstruction can be done in linear time by a counting argument, similar to the staircase polygons [1]. (We give such an algorithm in Section 4.5.2.) However, this construction relies on the symmetry of the two staircases and that nearly all vertices have unique degrees. This symmetry breaks down when moving to general histograms, where it is not clear that any counting strategy will work.

4.1 Overview of the Algorithm

Every histogram can be decomposed into axis-aligned rectangles whose contact graph is an ordered tree [11], as illustrated in Figure 7(b). In Section 4.2, we show that we can construct the (unordered) contact tree T from the visibility graph G_P in $O(n^2m)$ time by repeatedly "peeling" tabs from the histogram. We then show that each left-to-right ordering of T's k leaves (as well as a left-to-right orientation of the rectangles in the leaves) induces a histogram P'. For each candidate polygon P' (of $k!2^k$ candidates), we then compute its visibility graph $G_{P'}$ in $O(n \log n + m)$ time [17] and check if $G_{P'}$ is isomorphic to G_P. Instead of requiring an expensive graph isomorphism check [6], we show how to use the ordering of T to quickly test if G_P and $G_{P'}$ are isomorphic.

In Section 4.4 we show how to reduce the number of candidate histograms from $k!2^k$ to $(k-2)!2^{k-2}$, leading to the main result of our paper:

Theorem 2 *Given a visibility graph G_P of a histogram P with $k \geq 2$ tabs, we can reconstruct P in $O(n^2m + (k-2)!2^{k-2}(n \log n + m))$ time.*

Finally, we give a faster reconstruction algorithm when the histogram has a binary contact tree, solving these instances in $O(n^2m)$ time (Section 4.4).

4.2 Rectangular Decomposition and Contact Tree Construction

We construct the contact tree T from G_P by computing a set $\mathcal{T}$ of the k tab edges of G_P (Lemma 11). Each tab (u, v) is 1-simplicial and in a maximal 4-clique, since $\{u, v\} \cup (N(u) \cap N(v))$ is a 4-clique representing a unit square at the top of the histogram. Given the set $\mathcal{T}$ of tab edges, our reconstruction algorithm picks an edge t from $\mathcal{T}$ and removes the maximal 4-clique containing t. This is equivalent to removing an axis-aligned rectangle in P, and, equivalently, removing a leaf node from T. Moreover, it associates that node of T with four vertices of P: two *top vertices* that are convex and two *bottom vertices* that are either both reflex or are both convex *base* vertices. This process might result in a new tab edge, which we identify and add to $\mathcal{T}$.

4.2.1 Finding initial tabs

We start by finding the k tabs. Recall that every tab edge is 1-simplicial and in a maximal 4-clique. The converse is not necessarily true. Therefore, we begin by finding all 1-simplicial edges that are in maximal 4-cliques as a set of candidate edges, and later exclude non-tabs from the candidates.

Given a visibility graph $G_P = (V_P, E_P)$ of a histogram P and a maximal clique $C \subseteq V_P$, we call a vertex $w \in C$ an *isolated vertex with respect to P* if there exists a tab edge $(u, v) \in E_P$, such that $(N(u) \cup N(v)) \cap C = \{w\}$, i.e., of all vertices of C, only w is visible to some tab of P.

Lemma 10 *In a histogram, every 1-simplicial edge in a maximal 4-clique contains either a tab vertex or an isolated vertex.*

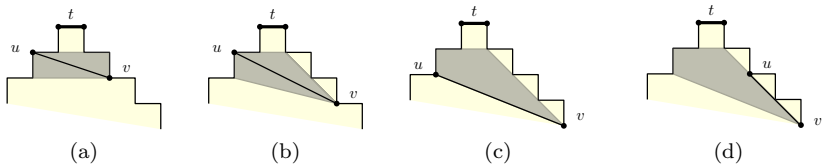


Figure 8: Edges between u and v on a tab t's staircases, but disjoint from t, are in a clique of size at least five.

Proof: Let ℓ_u denote the *level* of a vertex u in P, which is its y-coordinate assuming all uniform edges have unit length, where the base vertices are at level 0. Consider an arbitrary 1-simplicial edge (u, v) that is part of a maximal 4-clique C. We assume that neither u nor v is a tab vertex, otherwise we are done. Note that if $|\ell_u - \ell_v| \leq 1$ then u and v are in an axis-aligned rectangle in P defined by at least six vertices of G_P (see Figure 8(a)), i.e., contradicting the assumption that (u, v) is part of a maximal 4-clique.

Without loss of generality, suppose that u lies above v (i.e., $\ell_u - \ell_v \geq 2$) to the left of v (the case of u lying to the right of v, or below v, can be proven symmetrically). Since a top vertex does not see any vertices above it, v must be a bottom vertex. Thus, v sees a vertex of some tab t. We will show that either t cannot see any other vertex of C, or C must contain more than 4 vertices, contradicting the assumption that it is a 4-clique.

Let R be a set of reflex vertices on v's staircase on levels $\ell_v + 1$ up to and including ℓ_u. Observe, that v cannot be part of the maximal 4-clique that contains both vertices of t (since it just sees one of t's vertices), hence, we have that $1 \leq |R| \leq \ell_u - \ell_v$ ($|R| < \ell_u - \ell_v$ when u and v are on the staircases of different tabs and the vertices of t are below u).

Case 1: u and v belong to the staircases of a single tab t. (See Figure 8(b)-(d).) Consider the following cases:

1. u **is a top vertex:** (See Figure 8(b).) Then u and v must lie on different staircases and (u, v) belongs to a clique consisting of at least six vertices: vertex u, $(\ell_u - \ell_v) \geq 2$ reflex vertices on v's staircase, v, and u's two (reflex) boundary neighbors.

2. u **is a bottom vertex:** (See Figures 8(c)-(d).) Then (u, v) belongs to a clique consisting of at least five vertices: $(\ell_u - \ell_v + 1) \geq 3$ bottom vertices on v's staircase from levels ℓ_u through ℓ_v, and at least two vertices on the opposite staircase, e.g., the vertices of the vertical boundary edge spanning levels l_u and $l_u + 1$.

Case 2: Vertices u and v belong to the staircases of different tabs t' and t (see Figure 9). We call (u, v) a *crossing edge*. Consider the following cases:

1. u **is reflex:** Let (u, u') and (u, u'') be the vertical and horizontal boundary edges, respectively. Since u' and u'' do not see each other, if v sees both u'

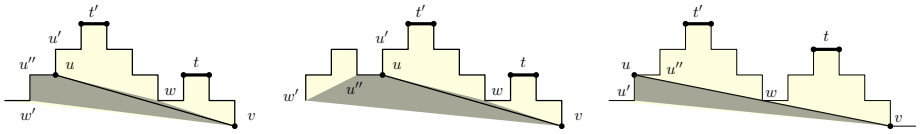


Figure 9: Illustration for the proof of Lemma 10. (a)-(b) If u is reflex, then the edge (u, v) is in a clique of size greater than four. (c) If u is convex and w blocks v from seeing u'', then v is an isolated vertex.

and u'', then (u, v) belongs to two cliques and, therefore, cannot be 1-simplicial. Since u'' is to the left of u, v sees u''. Therefore, there must be some vertex w that is visible to both u and v and that blocks the view from v to u'.

(a) u'' **is convex:** (See Figure 9(a).) Then consider the vertical boundary edge (u'', w'). Clearly, v sees w' and (u, v) belongs to a clique consisting of at least five vertices: $\{v, w, u, u'', w'\}$.

(b) u'' **is reflex:** (See Figure 9(b).) Then there must be at least one vertex $w' \in C$, bounding the convex region of C on the left (e.g., by line segments $\overline{u''w'}$ and $\overline{w'v}$). Again, v sees w' and (u, v) belongs to a clique consisting of at least five vertices: $\{v, w, u, u'', w'\}$.

2. u **is convex:** (See Figure 9(c).) Let (u, u') and (u'', u) be the vertical and horizontal boundary edges incident to u, respectively. Since v sees u and u' is below u, v must also see u'. Moreover, there must be some vertex (visible from both u and v), which bounds the convex region of C from the right. Let w be the closest such vertex to the edge (u, v). There are two cases to consider:

(a) w **is on** v**'s staircase:** Then v sees u'' and (u, v) belongs to a clique consisting of at least five vertices: $\{u, v, w, u', u''\}$.

(b) w **is not on** v**'s staircase:** If C is a 4-clique, u'' is not visible from v, i.e., w is below $\overline{vu''}$, and C consists of $\{u, v, w, u'\}$. Since w bounds the convex region from the right, it cannot be visible from t. Moreover, it also blocks u and u' from t's view. Thus, t sees only v among the vertices of C, i.e., v is an *isolated* vertex.

□

Lemma 11 *In a visibility graph of a histogram, tabs can be computed in time* $O(n^2m)$.

Proof: We begin by computing all 1-simplicial edges in maximal 4-cliques, which takes time $O(nm)$ by Observation 1. Call this set of edges E_{sim}, and the

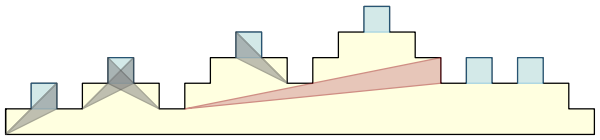


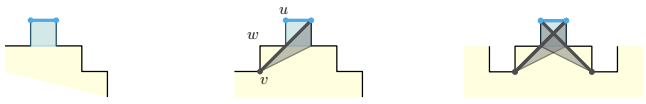
Figure 10: Illustrating all maximal 4-cliques that contain 1-simplicial edges. These include tab cliques (squares) and non-tab cliques (triangles). Note that tab cliques can share vertices with zero, one, or two non-tab cliques.

set of their maximal cliques $\mathcal{C}_{\text{sim}}$ (see Figure 10). Then E_{sim} contains the tabs, some edges that share a vertex with the tabs, and edges between staircases of different tabs (crossing edges) (which contain isolated vertices by Lemma 10). For all (non-incident) pairs of 1-simplicial edges e_1 and e_2 in maximal 4-cliques C_1 and C_2, respectively, we check if exactly one vertex of C_2 can be seen by an endvertex of e_1. That is, we compute the set $V_{12} = \{v \in C_2 \mid (u, v) \in E \text{ and } u \in e_1\}$ and verify that $|V_{12}| = 1$. If so, then C_2 is a non-tab clique and can be eliminated: If e_1 is a tab, then C_2 is a non-tab clique containing an isolated vertex (and a crossing edge); otherwise, e_1 is not a tab and its endvertices see either zero or three vertices of any tab clique, and therefore C_2 cannot be a tab clique. Thus, if we compare all pairs of edges and cliques, all 4-cliques containing crossing edges will be eliminated, leaving only tab cliques. We can do this check in $O(nm)$ time by first storing, for each vertex u, the edges $\{(u, v) \in E_{\text{sim}}\}$ and cliques $\{C \in \mathcal{C}_{\text{sim}} \mid u \in C\}$. Then for each edge (u, v) in G_P, we run the isolated vertex check for each pair of edges and cliques stored at the endvertices u and v. Each check of all pairs takes $O(n^2)$, and we do this for $|E_{\text{sim}}| = O(m)$ edges.

If only disjoint cliques remain after the previous step, then we have computed exactly all k tabs. Otherwise, we need to eliminate non-tab edges that share a tab vertex. Note that non-tab edges form cliques along the staircase incident to the tab. Since staircases in a histogram are disjoint, non-tab cliques only intersect where they intersect a tab clique. Therefore, the remaining set of cliques can be split into k mutually disjoint sets, where k is the number of tabs, and each set has at most three cliques that intersect (see Figure 10), of which exactly one is a tab clique, and the other at most two cliques contain a tab vertex, and its non-tab neighbors on the opposite staircase. Let S be one of these k sets. We can compute the tab in S as follows (three cases):

1. ($|S| = 1$) Then the tab vertices see fewer vertices than non-tab (reflex) vertices. We can see this as follows: tab vertices see the vertices in their tab clique, plus the bottom vertices on the tab's staircases. The non-tab reflex vertices see the same vertices, plus at least two other vertices at their same level, which tab vertices cannot see. See Figure 11(a).

2. ($|S| = 2$) The two cliques in S share three vertices, and two vertices u and v are in exactly one unique clique each. One of these vertices (say u) is on the tab, and sees fewer vertices than the other non-tab vertex (v)



(a) (b) (c)

Figure 11: Configurations of tab cliques (squares) overlapping non-tab 4-cliques (triangles) considered the case analysis in the proof of Lemma 11. Regions represent maximal 4-cliques. Emboldened lines are 1-simplicial edges. Blue regions (and lines) represent tab cliques (and tabs). (a) One (tab) clique. (b) Two overlapping cliques. (c) Three overlapping cliques.

does. We can see this as follows: assume without loss of generality that u is on the left staircase. Then u sees the three vertices of its tab clique, and all reflex vertices on the right staircase. Meanwhile v is on the same side (left) as u, and sees the same vertices as u, plus a (convex) boundary neighbor w, which is on the level between u and v, which u cannot see. The remaining tab vertex is adjacent to v along its 1-simplicial edge forming the clique in S. See Figure 11(b).

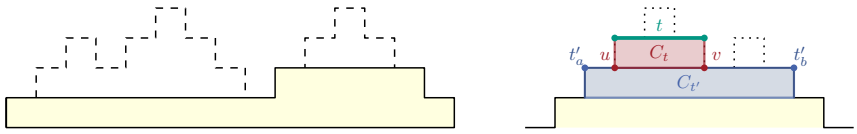
3. ($|S| = 3$) The cliques intersect symmetrically. The tab edge is formed between the two vertices that are in exactly two of these maximal cliques. See Figure 11(c).

After removing cliques with isolated vertices, there are at most $3k$ overlapping cliques in total, and they can be separated into k sets of cliques incident to each tab in $O(k)$ time by marking the vertices of each set, and collecting the intersecting sets. Then within each set, it takes $O(1)$ time to find the tab clique (and the tab). Thus, the running time is dominated by the time to detect crossing edges in E_{sim}: $O(n^2m)$. □

Note that top vertices cannot see the vertices above them. Therefore, only bottom vertices see tab vertices. Moreover, every bottom vertex sees at least one tab vertex. Thus, identifying all tabs immediately classifies vertices of G_P into top vertices and bottom vertices.

4.2.2 Peeling tabs

Let P' be a polygon resulting from peeling tab cliques (rectangles) from a histogram P. We call P' a *truncated* histogram. See Figure 12(a) for an example. After peeling a tab clique, the resulting polygon does not have uniform step length and the visibility graph may no longer have the properties on which Lemma 11 relied to detect initial tabs. Instead, we use the following lemma to detect newly created tabs during tab peeling.



(a) (b)

Figure 12: (a) A truncated histogram, created by iteratively removing six tabs (dashed) from a histogram. (b) When removing C_t: t'_a, t'_b form a tab if and only if $t'_a, t'_b \in T \setminus C_t$, they see u and v, and $|C_{t'}| = 4$.

Lemma 12 *When removing a tab clique from the visibility graph of a truncated histogram, any newly introduced tab can be computed in time $O(n)$.*

Proof: Denote the removed (tab) clique by C_t and let t be its tab. Let $u, v \notin t$ be the non-tab vertices of C_t. Since u sees v, (u, v) is an edge in G_P.

Since top vertices can only see vertices at and below their own level, besides the vertices of t, there are exactly two other top vertices in (remaining) G_P that see u and v, namely, the top vertices t'_a and t'_b of P on the same level as u and v (see Figure 12(b)). Since t'_a, t'_b are adjacent in G_P, let $t' = (t'_a, t'_b)$.

When removing C_t from G_P, we can compute t'_a and t'_b in time $O(n)$ by selecting the only two top vertices adjacent to both u and v. Since t'_a and t'_b are the top vertices of a same rectangle $R_{t'}$, edge t' is 1-simplicial and is in exactly one maximal clique $C_{t'} = N(t'_a) \cap N(t'_b)$, which corresponds to the convex region $R_{t'}$. Finally, after C_t is removed, t' is a newly created tab if and only if $|C_{t'}| = 4$, which can again be tested in time $O(n)$ by computing the common neighborhood $N(t'_a) \cap N(t'_b)$. □

With each tab clique (rectangle) removal, we iteratively build the parent-child relationship between the rectangles in the contact tree T as follows. Using an array A, we maintain references to cliques being removed whose parents in T have not been identified yet. When a tab clique C_t is removed from G_P, the reference to C_t is inserted into $A[u]$, where u is one of the rectangle's bottom vertices. If the removal of C_t creates a new tab $t' = (t'_a, t'_b)$, we identify $C_{t'}$ in $O(n)$ time using Lemma 12. Recall that t' sees all bottom vertices on the same level. Thus, for every bottom vertex $u \in N(t'_a)$ (in the original graph G_P), if $A[u]$ is non-empty, we set $C_{t'}$ as the parent of the clique stored in $A[u]$ and clear $A[u]$. This takes at most $O(n)$ time for each peeling of a clique. We get the following lemma, where the time is dominated by the computation of the initial tabs:

Lemma 13 *In $O(n^2 m)$ time we can construct the contact tree T of P, associate with each $v \in T$ the four vertices that define the rectangular region of v, and classify vertices of G_P as top vertices and bottom vertices.*

4.3 Mapping Candidate Polygon Vertices to the Visibility Graph

Let $\hat{T}$ correspond to T with some left-to-right ordering of its leaves and let $\hat{P}$ be the polygon corresponding to $\hat{T}$. We will map the vertices of G_P to the vertices of $\hat{P}$ by providing for each vertex of G_P the x- and y-coordinates of a corresponding vertex of $\hat{P}$. Let $t_1, t_2, \ldots, t_k$ be the order of the tabs in $\hat{P}$. Since $\hat{T}$ unambiguously defines the polygon $\hat{P}$, each node v of $\hat{T}$ is associated with a rectangular region on the plane, and the four vertices of G_P are associated with the four corners of the rectangular region. Since by Lemma 13 every vertex of G_P is classified as a top vertex or a bottom vertex, the y-coordinate can be assigned to all vertices unambiguously, because there are two top vertices and two bottom vertices associated with each node v of $\hat{T}$.

However, for every pair $p, \bar{p}$ of top vertices or bottom vertices associated with a node in $\hat{T}$ (call them *companion* vertices) there is a choice of two x-coordinates: one associated with the left boundary and one associated with the right boundary of the rectangular region. Thus, determining the assignment of each top vertex and bottom vertex in G_P to the left or the right boundary is equivalent to defining x-coordinates for all vertices in G_P. Although there appears to be $2^{n/2}$ possible such assignments, there are many dependencies between the assignments due to the visibility edges in G_P. In fact, we will show that by choosing the x-coordinates of the tab vertices, we can assign all the other vertices. Thus, in what follows we consider each of the 2^k possible assignments of x-coordinates to the $2k$ tab vertices.

At times we must reason about the assignment of a vertex to the left (right) staircases associated with some tab t_j. Given $\hat{T}$, the x-coordinates of each vertex in the left and right staircase associated with every tab t_j is well-defined. Therefore, assigning a vertex p to a left (right) staircase of some tab t_j defines the x-coordinate of p.

In a valid histogram, companion vertices p and $\bar{p}$ must be assigned distinct x-coordinates. Therefore, after each assignment below, we check the companion vertex and if they are both assigned the same x-coordinate, we exclude the current polygon candidate $\hat{P}$ from further consideration.

We further observe that in a valid histogram, if a bottom vertex p is not in the tab clique, then it sees exactly one tab vertex, which lies on the opposite staircase associated with that tab. Each bottom vertex can see through all other bottom vertices on its staircase along either a $45°$ or $135°$ angle to this tab vertex, and the bottom vertices of the tab clique block visibility with the other tab vertex. Thus, we can assign every bottom vertex the left (right) x-coordinate if it sees the right (left) tab vertex.

Next, consider any node v of the contact tree $\hat{T}$ and let R_v define the rectangle associated with v in the rectangular decomposition of a valid histogram. Let p be a top vertex in R_v and let $S(p)$ be the set of vertices visible from p that are not in R_v ($S(p)$ can be determined from the neighborhood of p in G_P). Observe that if p is assigned the left (right) x-coordinate, then every vertex in $S(p)$ is a bottom vertex to the right (left) of the rectangle R_v, none of them

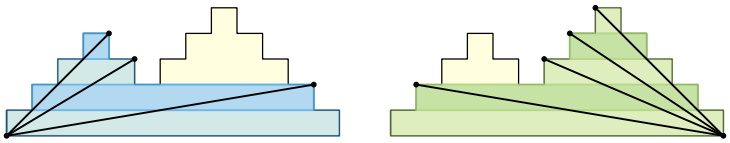


Figure 13: Visibility from the left (right) base vertex determines the left- (right-) most tab, and orients all rectangles on the left (right) spine of the contact tree.

belongs to a tab clique, and all of them are assigned a right (left) x-coordinate. Since the x- and y-coordinates of the boundaries of R_v are well-defined by $\hat{T}$ (regardless of vertex assignment), if $S(p)$ is non-empty, we check all of the above conditions and assign p an appropriate x-coordinate. If a condition is violated, then the current polygon candidate is invalid and we exclude it from further consideration.

Let p be one of the remaining top vertices without an assigned x-coordinate. If the companion $\bar{p}$ is assigned an x-coordinate, we assign p the other choice of the x-coordinate. Otherwise, both p and $\bar{p}$ see only the vertices inside their rectangle. In this case, the neighborhoods $N(p)$ and $N(\bar{p})$ are the same and we can assign p and $\bar{p}$ to the opposite staircases arbitrarily.

The only remaining vertices without assigned x-coordinates are bottom vertices in tab cliques. Consider a companion pair p and $\bar{p}$ of bottom vertices that are in a tab clique. Let R be the rectangle defined by the tab and let $S_{right}(p)$ (resp., $S_{left}(p)$) denote the set of vertices that p sees among the vertices to the right (resp., left) of R. Observe that if p is on the left boundary, then $S_{right}(\bar{p}) \subseteq S_{right}(p)$ or $S_{left}(p) \subseteq S_{left}(\bar{p})$. Symmetrically, if $\bar{p}$ is on the left boundary then $S_{right}(p) \subseteq S_{right}(\bar{p})$ or $S_{left}(\bar{p}) \subseteq S_{left}(p)$. Thus, if $|S_{right}(\bar{p})| \neq |S_{right}(p)|$ or $|S_{left}(p)| \neq |S_{left}(\bar{p})|$, we can assign p and $\bar{p}$ appropriate x-coordinates. Otherwise, the neighborhoods $N(p)$ and $N(\bar{p})$ are the same, and we can assign p and $\bar{p}$ to the opposite boundaries arbitrarily.

4.4 Reducing the Number of Candidate Histograms

We can reduce the number of possible orderings of tabs and staircases by considering only those that meet certain visibility constraints on the vertices that form the corners of each rectangle. In particular, we say that two rectangles $R_1 \neq R_2$ in the decomposition are *orientation-fixed* if a bottom vertex v_{bot} from one can see a top vertex v_{top} of another. Then these rectangles must be oriented so that v_{bot} and v_{top} are on opposite staircases (an up-staircase and a down-staircase). Thus, fixing an orientation of one rectangle fixes the orientation of the other.

Note that every rectangle is orientation-fixed with some leaf rectangle (as its bottom vertex can see a tab vertex). Therefore, ordering (and orienting) the leaves induces an ordering/orientation of the tree. There are $O(k!2^k)$ such orderings (and orientations) for all leaf rectangles, where k is the number of

tabs.

For double staircases, T is a path and the root rectangle is orientation-fixed with every other rectangle (every top vertex sees a base vertex). Hence, orienting the base rectangle determines the positions of the top vertices on the double staircase. Likewise, for the histogram, the spines of T are fixed:

Lemma 14 *The base rectangle of a histogram is orientation-fixed with all rectangles on the left and right spines of T.*

Moreover, the only tab vertices visible from a base vertex are incident to the left-most or right-most tab. Thus, we can identify the left-most and right-most tabs based on the neighborhood of the base vertices. Note that removing a base rectangle of the histogram produces one or more histograms. Then we can apply this logic recursively, leading to the following algorithm:

1. Fix the orientation of the base rectangle. This identifies the rectangles on the left and right spines of T and their orientations. (See Figure 13.)

2. The remaining subtrees collectively contain the remaining rectangles, which still must be ordered and oriented. We recursively compute the ordering and orientation of the rectangles in these subtrees.

Note if we compute the left and right spines of T, we identify the first and last tabs, and the orientations of their tab edges. Thus, we have $(k-2)!2^{k-2}$ remaining orderings of T and orientations of the tab edges to check, as $k-2$ tabs remain. This results in the overall reconstruction of a histogram with $k \geq 2$ tabs in $O(n^2m + (k-2)!2^{k-2}(n \log n + m))$ time, where $O(n^2m)$ is the time to compute the tabs of the histogram, and $O(n \log n + m)$ is the time to compute a candidate polygon's visibility graph (from an ordering of T and an orientation of the tab edges) and check it against the input graph. Thus, we have proved Theorem 2.

4.5 Polynomial-Time Algorithms for Special Cases

We now show that some histograms can be reconstructed in polynomial time: in particular, for histograms with a binary contact tree, and histograms with two staircases.

4.5.1 Histograms with a binary contact tree

We can generalize the technique from above to more accurately describe the number of candidate polygons, by giving a recurrence based on the structure of the contact tree. Let $v \in T$, and define $C(v)$ to be the set of v's children in T and $d(v) = |C(v)|$. Then if we have a fixed orientation of v's corresponding rectangle, fixing the rectangles on the left-most and right-most paths from v

limits the number of possible orderings/orientations of v's descendants to

$$F(v) \leq \begin{cases} 2^{d(v)-2} \prod_{u \in C(v)} F(u) & \text{if } d(v) > 1, \\ F(u) & \text{if } d(v) = 1, \text{ with } C(v) = \{u\}, \\ 1 & \text{if } d(v) = 0. \end{cases}$$

Note that $F(root) = 1$ when T is a binary contact tree. That is, the orientation of the base rectangle completely determines the histogram. Furthermore, we can find such an orientation by fixing the orientation of the base edge, determining the left- and right-most paths, ordering and orienting them to match the base edge, and then repeating this for each subtree whose root is oriented and ordered (but its children are not), which acts as a base rectangle for its subtree. This process can be done in time $O(n + m)$ by traversing T and orienting each rectangle exactly once by looking at its vertices' neighbors in its base rectangle in T. Here, the time is dominated by the time to compute the tabs: $O(n^2 m)$.

Theorem 3 *Histograms with a binary contact tree can be reconstructed in time* $O(n^2 m)$.

Note that, by Theorem 3, histograms consisting of two staircases can also be constructed in time $O(n^2 m)$ time. However, as we now show it is possible to construct them in linear time.

4.5.2 A linear-time algorithm for two staircases

We first note that in *double staircase* polygons (consisting of only two staircases) there is a simple linear-time reconstruction algorithm based on the degrees of vertices in the visibility graph.

Every double staircase polygon can be decomposed into k axis-aligned rectangles for some integer $k \geq 1$. Note that the number of vertices in such a polygon is $4k$, with each vertex u on one of $k + 1$ levels $l_u = 0, \ldots, k$. With the exception of levels $l_u = 0$ and $l_u = k$, which contain only two vertices (pairs of tab and base vertices), there are four vertices per level (two top and two bottom vertices).

Observe that the degree $deg(u)$ of each vertex $u \in G_P$ exhibits the following pattern:

(a) u_k **is a tab vertex (on level** k**).** $deg(u_k) = k + 2$: vertex u_k is a neighbor to k bottom vertices on the opposite staircase and two boundary vertices of u.

(b) u_l **is a top vertex on level** $l = 1, \ldots, k - 1$**.** $deg(u_l) = l + 4$: vertex u_l is a neighbor to $l + 1$ bottom vertices on the opposite staircase, 1 convex vertex on the same level as u_l on the opposite staircase, and two boundary vertices of u_l.

(c) u_l **is a bottom vertex on level** $l = 1, \ldots, k-1$**.** $deg(u_l) = 3k - l + 2$: vertex u_l is a neighbor to k bottom vertices on the opposite staircase, $k-l+1$ top vertices on the opposite staircase, $k-1$ bottom vertices on the same staircase and two boundary vertices of u_l.

(d) u_0 **is a base vertex (on level** 0**).** $deg(u_0) = 3k$: vertex u_0 is a neighbor to $2k$ vertices on the opposite staircase, $k-1$ bottom vertices on the same staircase, and one boundary vertex of u_0.

Observe, that most of the above values for degrees come in pairs: one vertex per staircase. For positive integer k, the only two exceptions are $deg(u) = k+2$ and $deg(u) = 3k$, each of which appears four times: two tab vertices u_k and u'_k and two top vertices u_{k-2} and u'_{k-2} (on level $k-2$), and two base vertices u_0 and u'_0 and two bottom vertices u_2 and u'_2 (on level 2). However, we can easily differentiate tab vertices u_k and u'_k from top vertices u_{k-2} and u'_{k-2}, because u_k and u'_k are neighbors to the two bottom vertices on level $k-1$ (of degree $2k+3$), while u_{k-2} and u'_{k-2} are not. Similarly, we can differentiate the two base vertices u_0 and u'_0 from bottom vertices u_2 and u'_2, because u_0 and u'_0 are neighbors to the two top vertices on level one (of degree 5), while u_2 and u'_2 are not.

Thus, we can identify the levels of each vertex by their degrees (or the degrees of their neighbors). Finally, an arbitrary left-right assignment of the two base vertices to the two staircases defines the assignment of all top (convex) vertices (including tab vertices) to the staircase. Furthermore, the assignment of tab vertices to the staircases defines the assignment of all reflex vertices to staircases. These assignments can be performed in linear time using the classification of the vertices into top and bottom vertices based on their degrees.

Theorem 4 *Double staircases can be reconstructed in* $O(n+m)$ *time.*

Finally, we briefly note that this is $O(n^2)$, since $k = n/4$, and the number of edges in a double staircase is

$$m = 3k + (k+2) + \sum_{l=1}^{k-1} [(3k - l + 2) + (l + 4)] = 3n^2/16 + 7n/4 - 4 = \Theta(n^2).$$

5 From Reconstruction to Recognition

Note that our reconstruction algorithms assign each vertex to a specific position in the constructed polygon, which we call a *vertex assignment reconstruction algorithm*. Assigning vertices to positions is not a requirement of reconstruction algorithms in general. However, as a result, the assignment enables us to develop recognition algorithms for these polygons as well. We first note the following.

Theorem 5 *If there exists a vertex assignment reconstruction algorithm* $\mathcal{A}$ *with time* $O(f(n))$ *for polygons in the class* $\mathcal{C}$*, and further if polygons in the class* $\mathcal{C}$ *can be recognized in time* $g(n)$*, then the visibility graphs of polygons in* $\mathcal{C}$ *can be recognized in time* $O(f(n) + g(n) + n \log n + m)$.

Proof: Given a graph G_P, run $\mathcal{A}$. If $\mathcal{A}$ fails to give a vertex assignment, or if it succeeds and the constructed polygon is not in the class $\mathcal{C}$, then G_P is not the visibility graph of a polygon in the class $\mathcal{C}$. If $\mathcal{A}$ successfully constructs a polygon P' in the class $\mathcal{C}$, then compute the visibility graph $G_{P'}$ of P' via the method of Ghosh and Mount [17], which takes time $O(n \log n + m)$ with respect to the output graph. Note that the number of edges of $G_{P'}$ could exceed that of G_P; however, we can stop the algorithm as the number of edges differ and report that G_P is not the visibility graph of a polygon in the class $\mathcal{C}$. Otherwise, if the number of edges is the same, we verify that all edges are between the same vertices. □

Uniform-length orthogonally convex *polygons* and histograms can be recognized the vertex coordinates (given in clockwise order) in time $O(n)$. Therefore, our reconstruction algorithms imply recognition algorithms for these polygon classes with the same running times. This also implies a recognition algorithm for double staircases in time $O(n \log n + m)$, which we note is not linear in the visibility graph size when $m = o(n \log n)$. However, there are several ways to correct this. Perhaps the simplest is to first check if the graph has $3n^2/16+7n/4-4$ edges. If not, we know it is not the visibility graph of a double staircase; otherwise $m = \Theta(n^2)$ and the remaining steps take $O(n \log n + m) = \Theta(m)$ time, which is linear in the graph size.

6 Conclusion and Future Work

In this paper, we showed how to reconstruct two classes of orthogonal polygons with uniform step lengths. For orthogonally convex polygons, we developed a reconstruction algorithm with $O(n^2 m)$ running time, and we further gave a fixed-parameter tractable reconstruction algorithm for histograms. Although we showed that histograms with a binary contact tree can be reconstructed efficiently, whether or not histograms can be reconstructed in polynomial time still remains open. Our original intent in studying histograms was to find a simple class of polygons that *may* be NP-hard to reconstruct. Their rigidity and integer coordinates ensure that histogram reconstruction is in NP, whereas it is unclear that polynomial-sized certificates exist for the general case. Furthermore, it would be interesting to see which *polyominoes*, which are a class of polygons formed by joining together unit squares along their edges, can efficiently be reconstructed.

Acknowledgments

We thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of this paper.

References

[1] J. Abello and Ö. Eğecioğlu. Visibility graphs of staircase polygons with uniform step length. *International Journal of Computational Geometry & Applications*, 3(1):27–37, 1993. doi:10.1142/S0218195993000038.

[2] J. Abello and K. Kumar. Visibility graphs of 2-spiral polygons (extended abstract). In R. Baeza-Yates, E. Goles, and P. V. Poblete, editors, *Proceedings of the 2nd Latin American Symposium on Theoretical Informatics (LATIN '95)*, volume 911 of *LNCS*, pages 1–15. Springer, 1995. doi:10.1007/3-540-59175-3_77.

[3] J. Abello and K. Kumar. Visibility graphs and oriented matroids. *Discrete & Computational Geometry*, 28(4):449–465, 2002. doi:10.1007/s00454-002-2881-6.

[4] J. Abello, Ö. Eğecioğlu, and K. Kumar. Visibility graphs of staircase polygons and the weak Bruhat order, I: From visibility graphs to maximal chains. *Discrete & Computational Geometry*, 14(3):331–358, 1995. doi:10.1007/BF02570710.

[5] T. Asano, S. K. Ghosh, and T. C. Shermer. Chapter 19–visibility in the plane. In J. Urrutia and J.-R. Sack, editors, *Handbook of Computational Geometry*, pages 829–876. North-Holland, Amsterdam, 2000. doi:10.1016/B978-044482537-7/50020-6.

[6] L. Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th ACM Symposium on Theory of Computing*, STOC '16, pages 684–697, 2016. doi:10.1145/2897518.2897542.

[7] L. Cai and H. Everett. Visibility graphs of polygonal rings. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 157–161, 1995.

[8] 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. doi:10.1007/BF01300372.

[9] P. Colley. Visibility graphs of uni-monotone polygons. Master's thesis, University of Waterloo, Waterloo, Canada, 1991.

[10] P. Colley. Recognizing visibility graphs of unimonotone polygons. In *Proceedings of the 4th Canadian Conference on Compututational Geometry*, pages 29–34, 1992.

[11] S. Durocher and S. Mehrabi. Computing partitions of rectilinear polygons with minimum stabbing number. In J. Gudmundsson, J. Mestre, and T. Viglas, editors, *Proceedings of the 18th International Conference on Computing and Combinatorics (COCOON 2012)*, volume 7434 of *LNCS*, pages 228–239. Springer, 2012. doi:10.1007/978-3-642-32241-9_20.

[12] H. ElGindy. *Hierarchical decomposition of polygons with applications*. PhD thesis, McGill University, Montreal, Canada, 1985.

[13] W. Evans and N. Saeedi. On characterizing terrain visibility graphs. *Journal on Computational Geometry*, 6(1):108–141, 2015. doi:10.20382/jocg.v6i1a5.

[14] H. Everett. *Visibility graph recognition*. PhD thesis, University of Toronto, 1990.

[15] H. Everett and D. Corneil. Recognizing visibility graphs of spiral polygons. *Journal of Algorithms*, 11(1):1–26, 1990. doi:10.1016/0196-6774(90)90026-B.

[16] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007. doi:10.1017/CBO9780511543340.

[17] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility. *SIAM Journal on Computing*, 20(5):888–910, 1991. doi:10.1137/0220055.

[18] M. Gibson, E. Krohn, and Q. Wang. A characterization of visibility graphs for pseudo-polygons. In N. Bansal and I. Finocchi, editors, *Proceedings of the 23rd European Symposium on Algorithms (ESA 2015)*, volume 9294 of *LNCS*, pages 607–618. Springer, 2015. doi:10.1007/978-3-662-48350-3_51.

[19] L. Jackson and S. Wismath. Orthogonal polygon reconstruction from stabbing information. *Computational Geometry: Theory and Applications*, 23 (1):69–83, 2002. doi:10.1016/S0925-7721(01)00068-2.

[20] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[21] J. O'Rourke and I. Streinu. Vertex-edge pseudo-visibility graphs: Characterization and recognition. In *Proceedings of the 13th Symposium on Computational Geometry*, SoCG '97, pages 119–128, 1997. doi:10.1145/262839.262915.