Reconstructing Polygons from Scanner Data[☆]

Therese Biedl[a,1], Stephane Durocher[b,1,*], Jack Snoeyink[c]

[a]*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada*
[b]*Department of Computer Science, University of Manitoba, Winnipeg, Canada*
[c]*Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, USA*

Abstract

A range-finding scanner can collect information about the shape of an (unknown) polygonal room in which it is placed. Suppose that a set of scanners returns not only a set of points, but also additional information, such as the normal to the plane when a scan beam detects a wall. We consider the problem of reconstructing the floor plan of a room from different types of scan data. In particular, we present algorithmic and hardness results for reconstructing two-dimensional polygons from point-wall pairs, point-normal pairs, and visibility polygons. The polygons may have restrictions on topology (e.g., to be simply connected) or geometry (e.g., to be orthogonal). We show that this reconstruction problem is NP-hard under most models, but that some restrictive assumptions do allow polynomial-time reconstruction algorithms.

Keywords: polygon, reconstruction, covering, algorithm

1. Introduction

Range scanners have been configured in many ways: looking in to capture objects on a platform or in-situ, looking down to capture terrain or urban environments, and looking out to capture rooms or factory floors. The problem of reconstructing surfaces in three dimensions from the resulting point clouds has been given both theoretical and practical consideration. Theoretical solutions can provably reconstruct the correct surface when the sample points are sufficiently dense relative to local feature size [1, 2, 7]. Applied solutions handle noisy data and often incorporate additional information such as estimated normals [19].

In addition to point coordinates, some scanners [6, 19] also return surface labels, normals, or the visible line segments from the scanner position, and the user may know something about the geometry (such as monotonicity and/or orthogonality) or topology

[☆]Some of these results appeared in preliminary form at the International Symposium on Algorithms and Computation [5].

[*]Corresponding author.

Email addresses: biedl@cs.uwaterloo.ca (Therese Biedl), durocher@cs.umanitoba.ca (Stephane Durocher), snoeyink@cs.unc.edu (Jack Snoeyink)

[1]Supported by NSERC.

(connectivity) of the environment. We originally sought to evaluate the utility of such information for the “simple” case of reconstructing the two-dimensional floor plan of a polygonal room. We discovered that even this two-dimensional problem is NP-hard for most models, and that we could achieve polynomial-time algorithms only for special cases.

1.1. Models and Problem Definition

We consider five models for input data that may be obtained by scanning a room with one or more scanners, as illustrated in Figure 1.

1. A *point scan* is a set of points, each of which lies on a wall (the interior of an edge) of the scanned room (polygon).
2. A *point-wall scan* is a point scan for which each point records the line containing the wall on which it lies (i.e., the orientation of the wall).
3. A *point-normal scan* is a point-wall scan for which each point-line pair records a normal perpendicular to the line that points towards the room’s interior. (This is a natural model for laser scanning systems that use multiple returns to reject noise, because they will typically create an estimate of the surface normal along with each point that they return.)
4. A *segment scan* is a point-normal scan for which each point records the position of its scanner; this implies that the entire line segment from a scanner to the corresponding scan point must be inside the room.
5. A *visibility-polygon scan* is a set of visibility polygons, i.e., the entire region visible from each scanner.

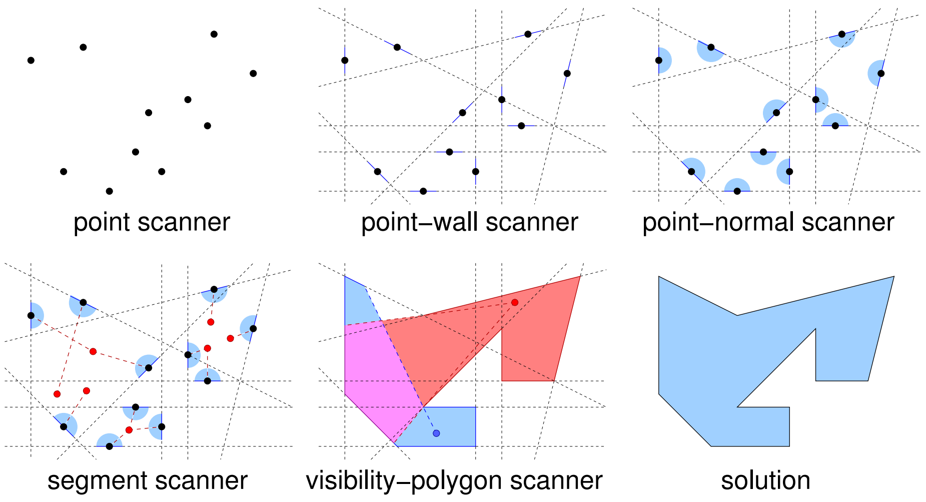

Figure 1: Input instances of the five scan models and a common solution.

Given n observations under of one of the five models, the polygon reconstruction problem is to determine whether there exists a polygon that is consistent with the input data. If there is, we would also like to know whether the solution is unique.

Without additional restrictions, the answer for the point-scan model is always "yes, a polygon exists and is not unique." One can easily construct a monotone polygon from any point set [23] in $\Theta(n \log n)$ time (many polygons, if the point set is not in convex position). The answer for other models is, therefore, also "yes" if a solution may include additional edges not encountered by any input point. In this paper, therefore, we assume that each wall has been seen, i.e., that each polygon edge contains at least one scan point. Unlike a number of previous polygon reconstruction algorithms in computational geometry [8, 9, 12, 20, 21, 23], in which scan points are polygon vertices, we require that each scan point lie in the *interior* of a polygon edge.

These five models define a proper hierarchy: any information available about a data point under one model is also available for all subsequent models. For example, the orientation of the wall at a data point provided in the point-wall scan model is also known in the point-normal, segment, and visibility-polygon scan models. Consequently, any polynomial-time algorithm for reconstructing a polygon under the point-wall model can also be applied to the point-normal, segment, and visibility-polygon scan models.

As we will see, however, all of these models are NP-hard, except in special cases with restricted environments. For polynomial-time reconstruction in these special cases we initially focused on the segment scan model, but we realized that our algorithms depended only on information about the orientation or normals of walls; hence, we formulate our algorithms for these simpler models.

1.2. Related Work

There are many previous results on reconstructing shape from densely-spaced samples on a surface in the point-scan model, including those cited earlier [1–3, 7, 19, 22]. Sampling density is critical in the point-scan model. For scan models that provide additional information, however, sample points need not be closely spaced if each edge includes at least one sample point. Thus, even with the caveat that sample points are drawn from edges, not vertices, our approach is closest to algorithms that reconstruct a polygon from its vertices. O'Rourke [20] gives an $O(n \log n)$ time algorithm for reconstructing an orthogonal polygon when edges form right angles at all vertices, and shows that when a solution exists, it is unique. This problem is NP-hard if edges at a vertex may meet either straight or at right angles [21] and also NP-hard if edges must be parallel to one of three (or more) given directions [12].

The reconstruction problem in many of our models can be formulated as a matching problem in a graph $G = (V, E)$ with additional restrictions. Each sample point corresponds to a segment on the polygon's boundary: let V contain two vertices for each sample point, one for each direction out from the segment. Join two vertices by an edge in E if the corresponding rays intersect. See Figure 2. The polygon reconstruction problem reduces to finding a spanning subgraph $H \subseteq G$ that has specific properties. In particular, we require that H be a perfect matching that is simple (no matching edges cross each other). Furthermore, if each pair of vertices induced by a sample point is joined by an edge, the resulting subgraph must be connected.

If the constraints of simplicity and connectivity are dropped, the problem is reducible to finding a perfect matching and is solvable in polynomial time [10, 13, 14, 17]. Adding either constraint renders the problem hard: finding a non-crossing 2-factor in a geometric graph was shown to be NP-hard by Jansen and Woeginger [16], and a connected 2-factor is simply a Hamiltonian cycle, which is well known to be NP-hard to find [13], even in grid graphs [15]. Neither of these results, however, directly implies hardness for the polygon reconstruction problems we consider.

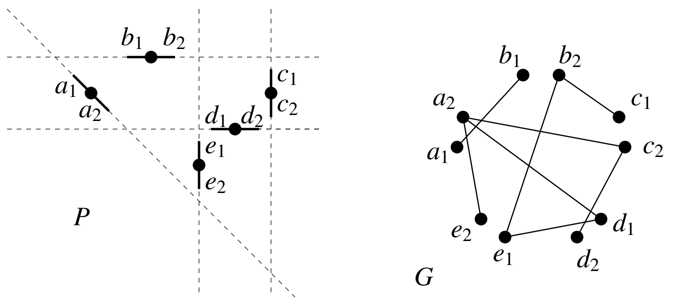

Figure 2: Under the point-wall scanner model, the line through each data point in set P corresponds to two rays, each of which can be represented by a vertex in graph G. Edges are added to G by connecting any two vertices in G whose corresponding rays intersect in P. Point set P has a (possibly crossing or disconnected) polygonal solution if and only if G has a perfect matching.

1.3. Our Results

We first show that the reconstruction problem is NP-hard, even when restricted to orthogonal edges (Section 2). We describe a reduction for the visibility-polygon scan model which we then generalize to the point-wall, point-normal, and segment scan models.

For positive results, we consider geometric restrictions to the allowable configurations of polygons. These geometric constraints may include requiring that a polygonal solution be star-shaped, monotone, or orthogonal. A *star-shaped polygon* is entirely visible from some point in its interior (i.e., the polygon can be seen by a single scanner). The interior of a *monotone*[2] *polygon* intersects every vertical line in at most one line segment. The boundary of a monotone polygon can be divided into two chains, the *upper* and *lower chains*, both of which are monotone. Finally, every edge in an *orthogonal polygon* is either horizontal or vertical.

Although our hardness reduction implies that the reconstruction problem remains NP-hard even for orthogonal polygons, we show that when both orthogonality and monotonicity are required, the problem can be solved in $O(n \log n)$ time under the point-wall scan model (Section 3). Similarly, we show that reconstruction is possible in $O(n \log n)$ time when monotonicity is required under the point-normal scan model (Section 4) or when a solution must be star-shaped under the point-normal scan model (Section 5). Finally, we present a lower bound showing that the running times of our algorithms are optimal (Section 6). See Table 1 for a summary of these results.

[2]For simplicity, we use the term *monotonicity* to refer to *x-monotonicity*.

	Unconstrained	Orthogonal	Monotone	Orthogonal Monotone	Star-Shaped
Point Wall	NP-hard	NP-hard	open	$\Theta(n \log n)$	polytime
Point Normal	NP-hard	NP-hard	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Segment	NP-hard	NP-hard	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Visibility Polygon	NP-hard	NP-hard	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$

Table 1: This table displays an overview of our hardness and algorithmic results for finding a polygonal solution under each combination of a given input model and a set of geometric constraints. In all cases we require that a solution be connected and non-crossing.

2. Hardness Results

In this section, we prove that reconstructing a simply-connected polygon from a visibility-polygon scan is NP-hard.[3]

We use a reduction from ORTHOGONAL NON-CROSSING SPANNING TREE, which was shown to be NP-hard by Jansen and Woeginger [16]. An *orthogonal graph* is a graph drawn in the plane such that every edge is a single horizontal or vertical line segment connecting two vertices and no edge contains any vertex in its interior. Edge crossings are allowed in the graph, but note that bends are forbidden, unlike some definitions used in the graph drawing literature. The task is to find a spanning tree of the vertices with no edge crossings.

ORTHOGONAL NON-CROSSING SPANNING TREE
Instance. An orthogonal graph G.
Question. Find a graph $H \subseteq G$ that is a spanning tree of G such that no two edges in H cross.

Theorem 1. *Polygon reconstruction under the visibility-polygon scan model is NP-hard.*

Proof. Given any orthogonal graph G, from an instance of ORTHOGONAL NON-CROSSING SPANNING TREE, we construct an instance of the visibility-polygon scan problem, $f(G)$, by replacing each vertex v in G with the vertex gadget $f(v)$ illustrated in Figure 3. In this gadget, there is a gap in the corresponding polygon edge for every neighbour of the vertex. This allows either connecting to the corresponding neighbouring vertex gadget via the corridor formed by a pair of parallel edges (blue, dashed), or closing off the gap by extending an edge (red, dotted). If a vertex has degree less than four, then the positions of edges near the corresponding scanners can be moved accordingly such that there is no gap (Figure 3B).

[3]The authors first presented the hardness results of Section 2 in a workshop abstract [4] and conference proceedings [5]. After this paper was accepted for publication, the authors learned of a similar result by Evrendilek et al. that appeared in the Proceedings of the 2010 International Symposium on Combinatorial Optimization [11].

Assume (after possible scaling) that G is drawn with vertices on the unit grid. Then $f(G)$ consists of a set of vertex gadgets such that a gadget of width and height $1/4$ is centered at the position of every vertex v in G. See Figures 3 and 4.

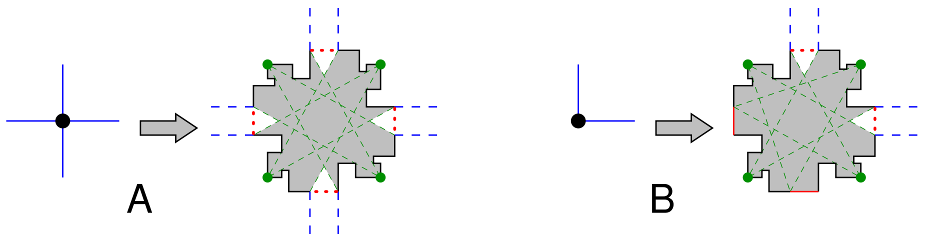

Figure 3: The vertex gadget is a portion of the polygon with four scanners: vertices of degree four (**A**) and degree two (**B**). Dashes indicate how a gadget may be closed or continued, provided it matches a corresponding gadget on the other end. Graph edge crossings that are not vertices need no gadget.

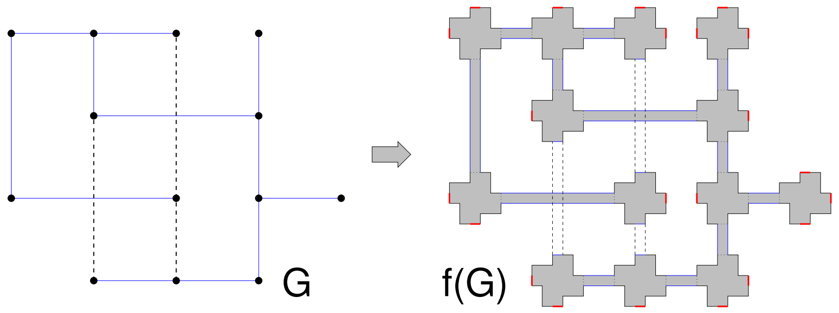

Figure 4: A spanning tree of a graph G and the corresponding simple polygon in $f(G)$

Components $f(v_1)$ and $f(v_2)$ can be joined by a pair of horizontal or vertical parallel edges forming a corridor if and only if vertices v_1 and v_2 are adjacent in G. Each edge in the corridor completes a partial edge in one of the two vertex gadgets. Note that the resulting instance $f(G)$ can be constructed in time proportional to the size of G on a Cartesian grid.

If G has a non-crossing spanning tree, then $f(G)$ has a simple polygonal solution formed by including the corridors that correspond to edges of the spanning tree. On the other hand, if $f(G)$ has a simple polygonal solution, then all vertex gadgets must be joined. Since every edge of the polygon must be seen by a scan, joining edges complete partial edges, and are therefore orthogonal. Since the polygon must be simple, the solution selects both or neither edge in a corridor, and edges from two crossing corridors cannot be selected simultaneously. See Figure 5. Therefore, G has a non-crossing spanning tree. □

Remark 1. Clearly, one can verify in polynomial time that the boundary and interior of every input visibility polygon agree with the boundary and interior of the solution polygon, that every edge of the solution polygon is met by an edge on the boundary of

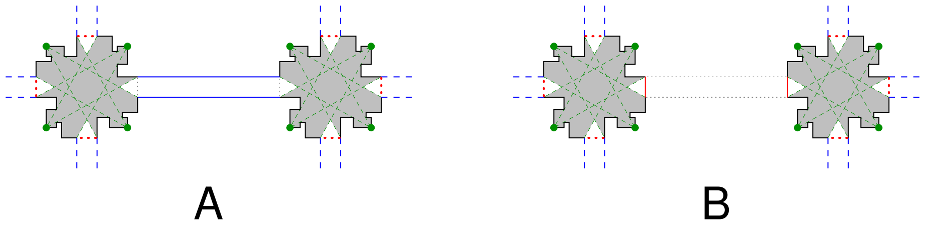

Figure 5: A pair of adjacent vertex gadgets may be joined by a pair of edges forming a corridor (**A**) or both endpoints of the corridor may be closed off, causing the gadgets to be locally disjoint (**B**). That is, either both or neither corridor edge can included in a solution.

some input visibility polygon, and that the solution polygon is non-crossing and connected. That is, the polygon reconstruction problem belongs to the set NP. Therefore, polygon reconstruction is NP-complete by Theorem 1.

Since all edges in the reduction are orthogonal, the visibility-polygon scan problem remains NP-hard for orthogonal polygons, giving the following corollary:

Corollary 2. *Reconstructing an orthogonal polygon under the visibility-polygon scan model is NP-hard.*

As we now show, the construction can be modified to show hardness for the point-wall scan, point-normal scan, and segment scan models.

Theorem 3. *Polygon reconstruction under the point-wall, point-normal, and segment scan models is NP-hard.*

Proof. The result follows by an argument analogous to that used to prove Theorem 1. Again, we use a reduction from ORTHOGONAL NON-CROSSING SPANNING TREE. Given any orthogonal graph G, we construct an instance of the point-wall scan problem, $f(G)$, by replacing each vertex v in G with the vertex gadget $f(v)$ illustrated in Figure 6A.

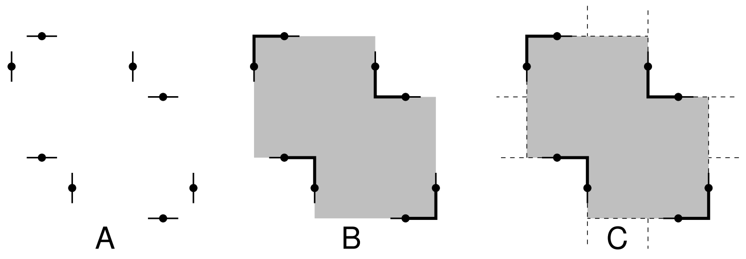

Figure 6: **A.** The vertex gadget under the point-wall scan model. **B.** Observation 1 implies the presence of the solid polygon edges in any solution. **C.** As in the proof of Theorem 1, adjacent vertex gadgets may connect via a pair of edges forming a corridor.

Under the visibility-polygon scan model, each vertex gadget's edges were visible to a scanner. Under the point-wall scan model, we must show that these edges in fact meet as desired in any solution. Given a scan point u met by a horizontal wall and a scan point v met by a vertical wall, let V, H, and VH denote the regions induced by the horizontal and vertical half-planes through u and v, as illustrated in Figure 7A. The following observation establishes a sufficient condition for u and v to be *adjacent*, i.e., for the edges through u and v to meet at a vertex of the polygon.

Observation 1. *If the interiors of regions H and VH are free of scan points met by horizontal edges and regions V and VH are free of scan points met by vertical edges, then u and v must be adjacent in every solution.*

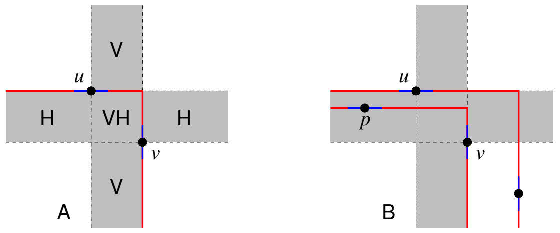

Figure 7: Illustration in support of Observation 1. **A.** Points u and v must be adjacent in any solution. **B.** The presence of point p in region H allows the possibility of a solution in which u and v are not adjacent.

Since scan points are positioned at grid coordinates, Observation 1 implies the existence of the solid edges in every vertex gadget, as illustrated in Figure 6B. When a vertex v in G has degree less than four, we modify the vertex gadget $f(v)$ accordingly to eliminate the possibility of including the corresponding corridor in any solution. This is achieved by adding two scan points to the corresponding side of the vertex gadget, as illustrated in Figure 8A. Edge e is forced; otherwise, edge f would be required, implying the existence of a vertical scan point that meets f. See Figure 8B. Any such scan point p is contained in a vertex gadget, say g_p; by Observation 1, p is constrained to lie on an edge joining its neighbours in vertex gadget g_p and, thus, cannot meet edge f.

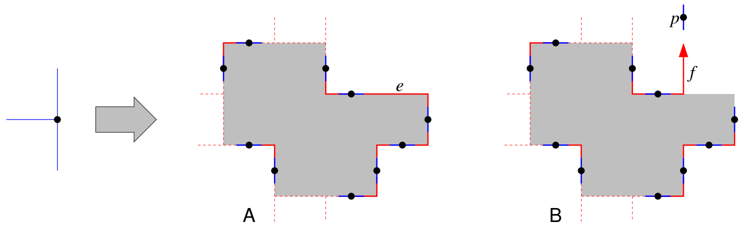

Figure 8: Since edge e is forced, this vertex gadget allows the possibility of connecting to only three corridors; the only possible obstruction would be an edge f coming from an eligible vertical scan point p in a neighbouring gadget g_p; by Observation 1, the edge through p is met by adjacent edges in g_p and, therefore, cannot meet f. A similar construction can be used to eliminate other corridors.

The remainder of the reduction is analogous to that described in the proof of Theorem 1. Similar reductions apply to the point-normal scan and segment scan models by using the corresponding vertex gadgets illustrated in Figure 9. The result follows. □

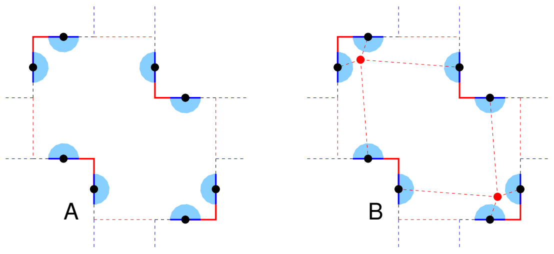

Figure 9: Vertex gadgets under the point-normal scan (**A**) and segment scan models (**B**).

Once again, the orthogonality of the construction implies the following corollary:

Corollary 4. *Reconstructing an orthogonal polygon under the point-wall, point-normal, and segment scan models is NP-hard.*

3. Orthogonal Monotone Polygons

Since the general reconstruction problem is NP-hard, we consider special cases that are solvable in polynomial time. Many actual room layouts are both orthogonal and monotone, motivating our consideration of these natural geometric constraints in this section. We show that an orthogonal monotone polygon can be reconstructed uniquely from a point-wall scan, i.e., each data point returns whether its edge is horizontal (H) or vertical (V).

Theorem 5. *A monotone orthogonal polygon can be reconstructed from a point-wall scan in $O(n \log n)$ time. Moreover, the solution is unique.*

Proof. We represent the input as a sequence σ of symbols over the alphabet $\{V, H\}$ in left-to-right order (breaking ties from bottom to top) in correspondence to whether the associated edge is vertical or horizontal. It will suffice to determine for each symbol whether it belongs to the lower or the upper chain; the chains are then easily reconstructed by parsing points in left-to-right order. This parsing may return that there is no feasible solution respecting the assignment to an upper/lower chain, but since (as we will see) the assignment to the chains is unique, this means that there was no feasible solution overall.

Our approach is to begin at a subsequence of σ for which the solution is uniquely determined locally, and then to propagate the solution, first to the right and then to the left. In all our claims below, we assume that the input can actually be realized by an orthogonal monotone polygon.

Claim 1. *Sequence σ contains a subsequence HH, i.e., two data points of horizontal edges with no vertical edge between their x-coordinates.*

Proof. Sequence σ must begin and end with V for the leftmost and rightmost edges (recall that no data point is at a vertex). Assume first that σ has no duplicate data points on vertical edges. Any orthogonal polygon has equally many horizontal and vertical edges, so σ has no more Vs than Hs. Since σ begins and ends with V, it hence contains HH.

Now assume σ has duplicate data points on vertical edges. Since points are sorted from left to right, this necessarily creates a VV; in other words, duplicate vertical data points can only duplicate existing Vs, not insert new ones. Let σ' be the substring of σ obtained by deleting all duplicate vertical data points. By the above argument, σ' contains HH, and since no V can be inserted between the two Hs by duplicate points, σ must also contain HH. $\square$

Claim 2. *At any subsequence HH, if both data points have the same y-coordinate, then they belong to the same edge. Otherwise the data point with larger y-coordinate must be in the upper chain and the other in the lower chain.*

Proof. Suppose the data points are labelled p_1 and p_2. Any vertical line with x-coordinate[4] between $p_1.x$ and $p_2.x$ must intersect both horizontal edges defined by p_1 and p_2. Thus if $p_1.y = p_2.y$, then this must be the same edge, otherwise the two chains would overlap. If $p_1.y \neq p_2.y$, then their order must determine which edge belongs to which chain. $\square$

Thus, search for an occurrence of HH in σ. If both points have the same y-coordinate, then remove one of them from σ (and later on, assign it to the same chain to which the other point was assigned). The resulting sequence must still contain HH. Continue searching until reaching an occurrence of HH whose points have different y-coordinates. This fixes two data points, say p_u and p_ℓ, to belong to horizontal edges of the upper and lower chains, respectively.

Now we extend the chains rightwards from p_u and p_ℓ. If the next element of σ is H, then it must be a duplicate data point for a horizontal edge. If it shares a y-coordinate with p_u or p_ℓ, then we assign it to the same chain and delete it from σ. Otherwise, there are three distinct horizontal edges without any vertical edge between them, which implies infeasibility of the input.

Now assume that the next element of σ is V; let p_v denote the corresponding data point. If the next element of σ is H (with corresponding data point denoted p_h), then the chains can be expanded as follows:

- If $p_v.y > p_u.y$, then p_v belongs to the upper chain (see Figure 10A).
- If $p_v.y < p_\ell.y$, then it belongs to the lower chain (see Figure 10B).
- If $p_\ell.y < p_v.y < p_u.y$, then the decision is determined by p_h:
  - If $p_v.y > p_h.y$ then p_v belongs to the upper chain (see Figure 10C).
  - If $p_v.y < p_h.y$ then p_v belongs to the lower chain (see Figure 10D).
- In all of the above cases, p_h belongs to the same chain as p_v.

[4]For all $a \in \mathbb{R}^2$, let $a.x$ and $a.y$ denote the respective x- and y-coordinates of a.

- In all other cases ($p_v.y = p_u.y$ or $p_v.y = p_\ell.y$ or $p_v.y = p_h.y$) there is either a crossing between the chains or a data point at a vertex, so we break and declare the input infeasible.

Correctness of these steps is straightforward; we only argue the first case here. Assume $p_v.y > p_u.y$. If p_v belonged to the lower chain, then the edge through p_u (which then would extend beyond $p_v.x$) would intersect the lower chain (containing p_ℓ and p_v). Therefore, if a feasible solution exists, then p_v belongs to the upper chain.

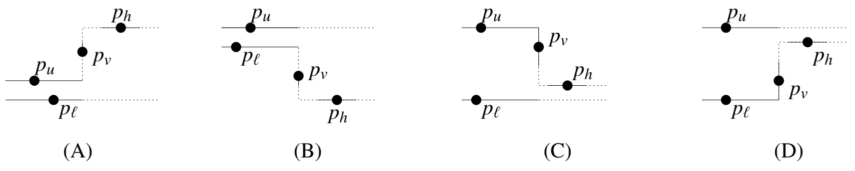

Figure 10: Four cases for resolving a substring *VH*.

Thus, if the next two elements of σ are *VH*, we can resolve these two edges. Moreover, again we know the y-coordinates of the last horizontal edge of the upper and lower chains, and hence can repeat the process.

If we reach *VV*, the scanning may stop. Let $p_1, p_2, \ldots, p_k$ be the next k data points that all are vertical ($k \geq 2$; $k > 2$ is feasible only if there are multiple data points on a vertical edge), and let p_h be the (horizontal) data point thereafter. We proceed as follows:

- If there are three or more x-coordinates among $p_1, \ldots, p_k$, then no realizing polygon can exist.
- If there are exactly two x-coordinates among $p_1, \ldots, p_k$, then this determines two distinct vertical edges. Delete all data points except p_1 and p_k.
- If $p_1, \ldots, p_k$ all have the same x-coordinate, then use their y-coordinates as well as p_h to determine which of them belong to the same vertical edge.
  - If $p_1.y < p_\ell.y < p_k.y$, then p_1 and p_k cannot both belong to the lower chain, and so there are two different vertical edges among these data points. See Figure 11A. Delete $p_2, \ldots, p_{k-1}$.
  - If $p_1.y < p_u.y < p_k.y$, then similarly p_1 and p_k are in different vertical edges. Delete $p_2, \ldots, p_{k-1}$.
  - If $p_\ell.y < p_1.y < p_k.y < p_u.y$, then let $0 \leq i \leq k$ be such that $p_i.y < p_h.y < p_{i+1}.y$. Then $p_1, \ldots, p_i$ must all belong to one vertical edge, and $p_{i+1}, \ldots, p_k$ belong to another. See Figure 11B.
    - ∗ If $i = 0$ or $i = k$, delete $p_2, \ldots, p_k$. This leaves a subsequence *VH* in σ, which we resolve as explained earlier.
    - ∗ If $1 \leq i < k$, delete $p_2, \ldots, p_k$.

- We are now left with two vertical data points p_1, p_k that are known to belong to different vertical edges. There is no horizontal edge between them, so they must belong to two different chains.
- From the y-coordinates of p_ℓ, p_u, p_1 and p_k, we can now determine which of p_1 and p_k belongs to the lower/upper chain.
  - If $p_1.y < p_\ell.y$, then p_1 belongs to the lower chain and p_k to the upper one. See Figures 11C1 to 11C3.
  - If $p_1.y > p_u.y$, then p_1 belongs to the upper chain and p_k to the lower one.
  - If $p_\ell.y < p_1.y < p_u.y$:
    * If $p_1.y < p_k.y$, then p_1 belongs to the lower chain and p_k to the upper one. See Figures 11D1 and 11D2.
      To see the correctness in this case, note that the chain containing p_1 must continue with a horizontal edge that extends beyond $p_k.x$. If p_k belonged to the lower chain, then this horizontal edge would have y-coordinate $< p_1.y$, and therefore the path through p_1 and this horizontal edge would cross the path through p_ℓ and p_k.
    * If $p_1.y > p_k.y$, then similarly p_1 belongs to the upper chain and p_k to the lower one.

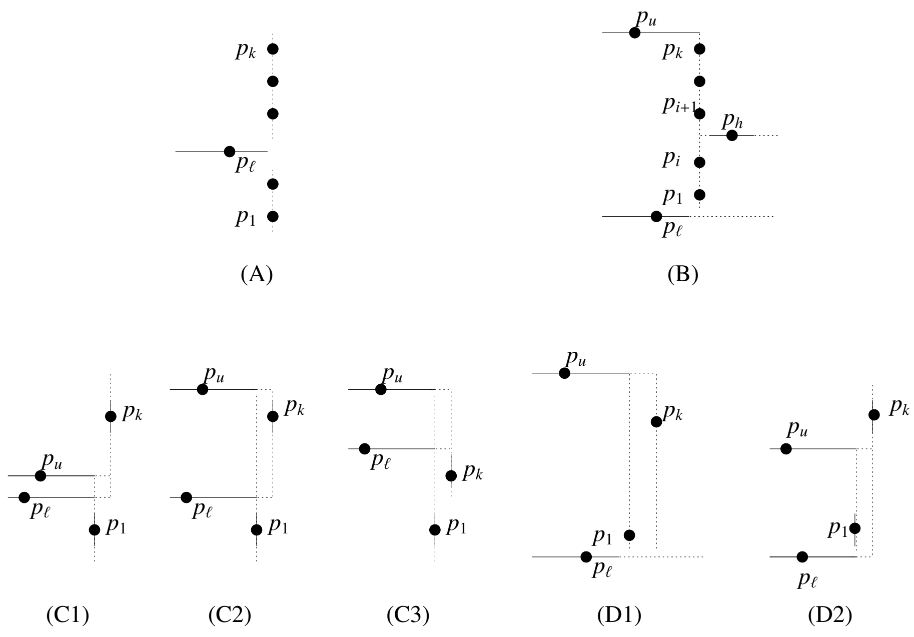

Figure 11: Two cases for resolving vertical data points with the same x-coordinates, and two cases for resolving a substring VV.

We cannot continue the scan beyond this *VV*, since no y-coordinates of the upper and lower chains are known to the right. But we can continue the scan somewhere farther to the right:

Claim 3. *Let $\sigma = \sigma_1\sigma_2$, where σ_1 ends with VV and these two Vs represent distinct vertical edges. Then σ_2 contains HH.*

Proof. The proof is similar to that of Claim 1. If σ_2 contains no two data points on a vertical edge, then $V\sigma_2$ (which could be completed to an orthogonally monotone polygon) contains at least one *HH*. Adding duplicate data points on vertical edges cannot insert a *V*, so the original σ_2 must also contain *HH*. □

Thus if we cannot continue the scan at a substring *VV*, then there must be a substring *HH* later on. We jump forward to this occurrence of *HH*, and then resolve rightward from then on until we reach another *VV*, jump forward to the next *HH*, and so on. This continues until at some point we reach a subsequence of *V*s that is not followed by *H*; we have then reached the rightmost edge.

We then repeat the same process in the opposite direction: start at the rightmost *HH*, resolve each substring *HV* leftward (in a symmetric manner) until we reach *VV*, eliminate duplicate vertical points at this *VV*, continue if possible, else jump leftward to the next *HH*, and so on, until we have reached the leftmost edge.

The process stops at a *VV* substring only if this represents two distinct vertical edges. As in Claim 3, one argues that if the rightward scan stops at a *VV*, and the leftward scan stops at a later *VV*, then they must have a substring *HH* between them. Thus no gaps remain between such *VV* stopping points. Upon terminating, we have determined for every edge whether it belongs to the upper or lower chain.

The time complexity of the algorithm is linear once the data points are sorted by x-coordinates. Furthermore, the resulting orthogonal polygon is unique since each edge is deterministically assigned to a chain. □

4. Monotone Polygons

In this section we consider the reconstruction problem for monotone (not necessarily orthogonal) polygons from a point-normal scan. In this case, each input point knows the orientation and interior of the polygon boundary passing through it. In the monotone setting, these half-planes determine whether each non-vertical edge belongs to the upper or lower chain of the polygon. This leaves the set of vertical edges to be assigned to chains. Nevertheless, the problem is non-trivial; in particular, a solution is not necessarily unique (e.g., see Figure 12).

Theorem 6. *A monotone polygon can be reconstructed from a point-normal scan in $O(n \log n)$ time.*

Proof. We use a dynamic programming algorithm that determines the chain to which vertical edges are assigned. Scan all data points from left to right and update a function that stores whether there is a partial solution (in the form of an upper and lower chain) up to the current x-coordinate, with some conditions on where the upper and lower chains end.

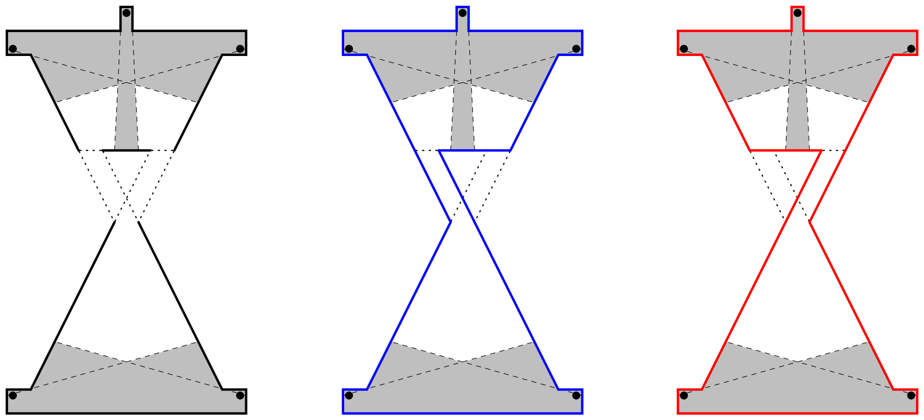

Figure 12: Even under the visibility-polygon scan model this input instance has two distinct monotone solution polygons; the central horizontal edge could belong to the left or right chains. Note, the figures are rotated by 90°, i.e., they are y-monotone; our algorithm refers to x-monotone polygons and upper or lower chains.

We introduce additional definitions to simplify references to points on chains. Let t be an x-coordinate that is not the coordinate of any data point. The *upper-left line* of t is the line through the last data point before t for which the edge is not vertical and is in the upper chain (i.e., the associated normal points downward). The *upper-right line* of t is the line through the first data point after t for which the edge is not vertical and is in the upper chain. We define the *lower-left* and *lower-right* lines of t analogously.

Observe that the vertical line through t intersects the upper chain of any solution necessarily in either the upper-left line or the upper-right line; otherwise one of the corresponding data points could not be used for the upper chain (and could not be used for the lower chain by the given normals.) We compute a partial solution and prescribe which of the two lines it uses for the upper chain, and correspondingly for the lower. Thus, define $f(t, u, \ell) \in \{\text{true}, \text{false}\}$, where $u, \ell \in \{L, R\}$, with $f(t, u, \ell) = \text{true}$ if and only if there exist two monotone chains such that

- the two chains have a common left endpoint point and do not intersect each other (hence they define an upper chain and a lower chain, respectively),
- the upper chain ends at t using the upper-left line if $u = L$, and using the upper-right line if $u = R$,
- the lower chain ends at t using the lower-left line if $\ell = L$, and using the lower-right line if $\ell = R$,
- the two chains use all lines through data points to the left of t, with the half-planes on the correct side, and use no other lines except the upper-right or lower-right line if so indicated by u or ℓ.

See also Figure 13.

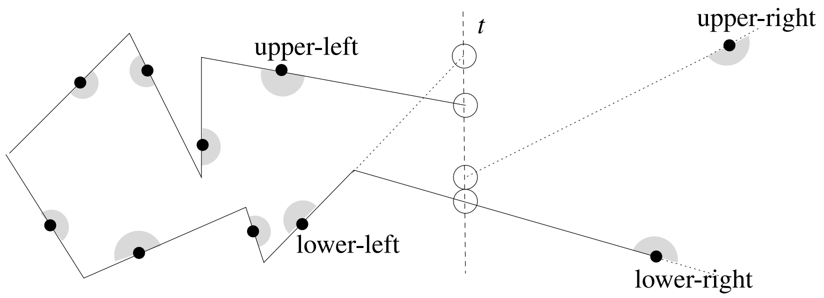

Figure 13: In this example $f(t, L, R) = \text{true}$. White circles indicate points in which the upper/lower chain might be required to end.

.

We can initialize $f(t, u, \ell)$ at the leftmost point of the polygon and update $f(t, u, \ell)$ as t increases. More precisely, to initialize, compute the upper-right and lower-right lines with respect to $-\infty$. If the leftmost data point does not lie on a vertical line, then initialize t to be the x-coordinate of the intersection of these two lines, and set $f(t + \varepsilon, R, R) = \text{true}$.[5] If the leftmost data point has a vertical line, then initialize t to be the x-coordinate of that data point, and initialize $f(t + \varepsilon, R, R) = \text{true}$ if and only if the data point lies below the upper-right and above the lower-right lines on the vertical line $\{x = t\}$. All other values $f(t + \varepsilon, u, \ell)$ are initialized to be false.

We update $f(t, u, \ell)$ as t increases. The truth assignment of $f(t, u, \ell)$ may change only at a value of t that is the x-coordinate of a data point, or the x-coordinate of a crossing of two of the four "adjacent" lines at t (because the chains might then also cross). Most of these updates are quite straightforward and are omitted here; we only describe the update for one of the complicated cases.

Suppose a data point has x-coordinate t, is not the rightmost data point, and lies on a vertical line with normal to the left. If the upper chain uses this vertical edge, then the order along line $\{x = t\}$ from bottom to top must be "intersection point with upper-right line," "data point for vertical line," and "intersection point with upper-left line." See Figure 14. Furthermore, the lower chain must not interfere with this, i.e., depending on the value of ℓ, the lower-left or the lower-right line must intersect $\{x = t\}$ below the upper-right line.

Similarly we can determine from the equations of the four lines adjacent to t whether the lower chain could use this vertical segment. Now we update according to the following boolean expressions:

- $f(t + \varepsilon, R, L) = f(t - \varepsilon, L, L)$ AND the upper chain could use the vertical edge.
- $f(t + \varepsilon, L, R) = f(t - \varepsilon, L, L)$ AND the lower chain could use the vertical edge.

[5]"$t + \varepsilon$" means "for values greater than t and smaller than the next x-coordinate where f may change. Similarly we use $t - \varepsilon$. In reality, parameter t is not needed since we always update from $t - \varepsilon$ to $t + \varepsilon$; we use it here to simplify notation.

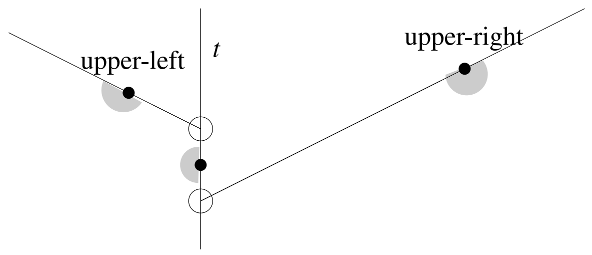

Figure 14: Using a vertical left-facing data point for the upper chain.

- $f(t+\varepsilon, R, R) = (f(t-\varepsilon, L, R)$ AND the upper chain could use the vertical edge) OR ($f(t-\varepsilon, R, L)$ AND the lower chain could use the vertical edge).
- $f(t+\varepsilon, L, L) = \text{false}$. (One of the two chains must use the vertical segment.)

This update, as well as all updates in all other cases, can be done in constant time. The time complexity for this algorithm is linear once the data points have been sorted by x-coordinate, resulting in a total running time of $O(n \log n)$. □

5. Star-Shaped Polygons

We briefly describe simple results for reconstructing a star-shaped polygon, i.e., a polygon that is entirely visible from some point in its interior. The region of points that see all of a star-shaped polygon is its *kernel*.

Reconstructing a star-shaped polygon is straightforward in the point-normal model. Any point in the kernel must be visible to all data points. It suffices to verify that the intersection of the set of half-planes associated with data points is non-empty, and to identify a point o in the interior of the intersection. This can be achieved in $O(n)$ time using the linear-programming algorithm of Megiddo [18]. If no such point o exists, then there is no solution. To compute the solution, sort all data points in clockwise order around o in $O(n \log n)$ time, and compute the polygon defined by them in this order. Either this polygon is star-shaped or there is no solution. Consequently, a solution is unique if it exists, giving the following theorem:

Theorem 7. *A star-shaped polygon can be reconstructed from a point-normal scan in $O(n \log n)$ time. Moreover, the solution is unique.*

We can also reconstruct a star-shaped polygon from a point-wall scan, but the time complexity increases. Consider the arrangement defined by the set of lines that pass through walls. As for point-normal scans, the kernel of a star-shaped polygon must be one of the cells defined by this arrangement, i.e., one of the maximal connected regions that do not contain a point on a line. There are $O(n^2)$ such cells for n lines. For each cell we can select a point o and attempt to reconstruct a star-shaped polygon with o in its kernel as explained above. The corresponding time complexity is $O(n^3 \log n)$, giving the following theorem:

Theorem 8. *A star-shaped polygon can be reconstructed from a point-wall scan in polynomial time.*

We suspect that the running time can be improved: instead of repeating the $O(n \log n)$ test in every cell, it might be possible to update the intersection of half-planes dynamically each time a half-plane is crossed. Furthermore, we believe that the solution, if one exists, is unique. Both of these questions remain open.

6. Lower Bound

We show the following lower bound on the worst-case running time of any algorithm that finds an orthogonal solution to an instance of the polygon reconstruction problem under the point scan, point-wall scan, point-normal scan, or segment scan models. This lower bound also applies to the orthogonal monotone case, showing that our results in Theorems 5, 6 and 7 are optimal.

Theorem 9. *Any algorithm that reconstructs an orthogonal polygon from point scans requires $\Omega(n \log n)$ comparisons in the worst case. Any algorithm that reconstructs a polygon from point-wall scans, point-normal scans, or segment scans requires $\Omega(n \log n)$ comparisons in the worst case. Furthermore, these lower bounds also apply to the cases for which a solution must be orthogonal, monotone and star-shaped.*

Proof. We describe a linear-time reduction from sorting, for which we assume that an input instance consists of n distinct integers $X = \{x_1, \ldots, x_n\}$. Let $x_{\min}$ and $x_{\max}$ denote the respective minimum and maximum values in X; these values can be found in linear time. Now define a set P of scan points as follows. For each $x_i \in X$ add points $(2x_i, 2x_i)$ and $(2x_i + 1, 2x_i + 1)$ to P. Furthermore, add points $(2x_{\min} + 1, 2x_{\min} - 1)$ and $(2x_{\max} + 2, 2x_{\max})$ to P. See Figure 15.

We claim that there exists a unique orthogonal polygon solution that realizes the points in P under the point scan model. The four extreme points must be as in Figure 15 in any realizing polygon. This forces $(2x_{\min} + 1, 2x_{\min} + 1)$ to be realized with a horizontal edge, for if it were vertical there would be no (unused) data point for a horizontal edge at its lower end. For similar reasons, this then forces $(2x', 2x')$ to be horizontal, where x' is the next-smallest number in X. Continuing this argument shows that all data points must have edges with the orientation as in Figure 15. Now applying Observation 1 shows that these data points can only be realized by the given polygon. Note that this polygon is monotone and star-shaped.

This solution gives the points of X in sorted order while walking along the polygon boundary. Consequently, reconstruction from a point scan sorts the values in set X and, therefore, requires $\Omega(n \log n)$ comparisons in the worst case.

If we endow the points with corresponding horizontal/vertical walls, or normals, or place a scanner at $(2x_{\max} + 1, 2x_{\min})$, then the same lower bound holds for the point-wall scan, point-normal scan and segment-scan models. □

We can also create a similar lower bound for the visibility-polygon model; the resulting polygon is orthogonal and monotone, but not star-shaped.

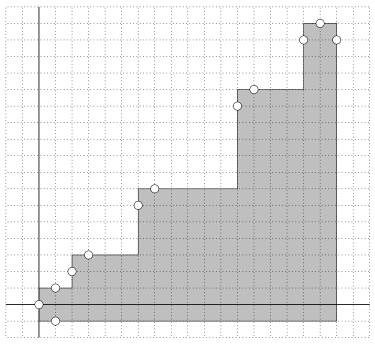

Figure 15: Given the set $X = \{0, 1, 3, 6, 8\}$, the transformation returns this set of points P and the corresponding orthogonal (monotone and star-shaped) polygonal solution.

Theorem 10. *Any algorithm that reconstructs a polygon from visibility-polygon scans requires* $\Omega(n \log n)$ *comparisons in the worst case. Furthermore, this lower bound also applies to the cases for which a solution must be orthogonal and monotone.*

Proof. As in the proof of Theorem 9, we describe a linear-time reduction from sorting. Again, suppose an input instance consists of n distinct integers $X = \{x_1, \dots, x_n\}$ and let $x_{\min}$ and $x_{\max}$ denote the respective minimum and maximum values in X.

We define a set S of scanners as follows. For each $x_i \in X$ add scanners at $(4x_i, -7)$ and $(4x_i, 7)$. Each such scanner is placed within a unit-width vertical tunnel, closed at one end and open at the other, such that the open ends of opposing scanners face each other. These corridors have the property that a scanner can see only edges in its corridor and the corridor immediately above/below it. See Figure 16A. Finally add two scanners at $(-z, 0)$ and $(z, 0)$, where z is a sufficiently large integer ($z = \max\{6x_{\max} - 2x_{\min}, 2x_{\max} - 6x_{\min}\}$ suffices). These two scanners are placed within unit-height horizontal tunnels such that, again, each can see only edges in its corridor and the corridor at the opposite left/right end. See Figure 16B.

Clearly a solution can be realized that meets the scanners in order of x-coordinates. Every edge in a solution must be visible by some scanner. Consequently, each incomplete edge segment in our construction has only a single possible edge with which it can be paired. It follows that the solution is unique. Observe that the solution polygon is orthogonal and monotone. The reconstructed polygon gives the points of X in sorted order while walking along the polygon boundary. Consequently, reconstruction from a visibility-polygon scan sorts the values in set X and, therefore, requires $\Omega(n \log n)$ comparisons in the worst case. □

7. Discussion and Directions for Future Research

We have examined the problem of polygon reconstruction from scanner data under various models of scanner input. As shown, the problem is NP-hard, and remains NP-hard even if a solution is required to be orthogonal. This hardness motivated our

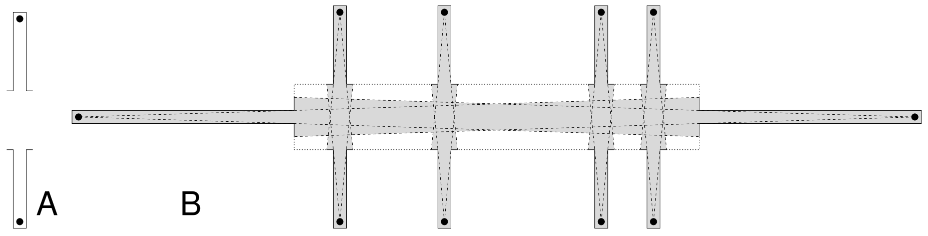

Figure 16: **A.** The placement of a pair of opposed scanners, illustrating the visible edges within their respective vertical tunnels. **B.** Given the set $X = \{1, 3, 6, 7\}$, the transformation returns this set S of scanners (marked by black points) and the corresponding unique orthogonal monotone polygonal solution. The placement of the leftmost and rightmost scanners is not to scale.

examination of polynomial-time algorithms when specific geometric constraints are imposed on a solution, including monotonicity, orthogonality, and/or star-shapedness.

If a solution is not unique, a natural question is to determine the number of additional scanners necessary to reveal the true solution. This question is NP-hard since Theorem 1 shows hardness for an instance of the corresponding decision problem. Approximation algorithms might be interesting to consider.

Several variants of our problem have not yet been considered and remain open. In particular:

- Can we reconstruct a monotone polygon from a point-wall scan? This is the only remaining unresolved complexity question in Table 1.
- What other restrictions on a solution make reconstruction feasible in polynomial time? For example, a reasonable assumption could be that a room has four walls, each of which is a polygonal chain: two x-monotone walls and two y-monotone walls.
- Finally, a natural question is to consider the corresponding problems in three or higher dimensions.

Acknowledgements

The authors wish to thank Joseph Mitchell for suggesting that we examine lower bounds (Section 6) as well as a possible technique for improving the running time of our algorithm for star-shaped polygons under the point-wall scan model (Section 5).

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.

[2] N. Amenta, S. Choi, and R. Kolluri. The power crust, union of balls and the medial axis. *Computational Geometry: Theory and Applications*, 19(2–3):127–153, 2001.

[3] F. Bernardini, C. L. Bajaj, J. Chen, and D. R. Schikore. Automatic reconstruction of 3D CAD models from digital scans. *International Journal of Computational Geometry and Applications*, 9:327–369, 1999.

[4] T. Biedl, S. Durocher, and J. Snoeyink. Reconstructing polygons from scanner data. In *Abstracts of the Fall Workshop on Computational Geometry*, volume 18, pages 51–52, 2008.

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

[6] DeltaSphere, Inc. Deltasphere 3D laser scanner, 2001. www.deltasphere.com/DeltaSphere-3000.html.

[7] T. K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge, 2007.

[8] S. Durocher. Graph theoretic and geometric algorithms associated with moment-based polygon reconstruction. Master's thesis, University of British Columbia, 1999.

[9] S. Durocher and D. Kirkpatrick. On the hardness of turn-angle-restricted rectilinear cycle cover problems. In *Proceedings of the Canadian Conference on Computational Geometry*, volume 14, pages 13–16, 2002.

[10] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[11] C. Evrendilek, B. Genç, and B. Hnich. Covering oriented points in the plane with orthogonal polygons is NP-complete. In *Proceedings of the International Symposium on Combinatorial Optimization*, volume 36, pages 303–310. Electronic Notes in Discrete Mathematics, 2010.

[12] M. Formann and G. J. Woeginger. On the reconstruction of simple polygons. *Bulletin of the European Association for Theoretical Computer Science*, 40:225–230, 1990.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, 1979.

[14] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 1(4):472–484, 1988.

[15] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[16] K. Jansen and G. J. Woeginger. The complexity of detecting crossingfree configurations in the plane. *BIT*, 33(4):580–595, 1993.

[17] L. Lovász and M. D. Plummer. *Matching Theory*, volume 29. North-Holland Mathematics Studies, 1986.

[18] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, 1984.

[19] L. Nyland, A. Lastra, D. McAllister, V. Popescu, and C. McCue. Capturing, processing and rendering real-world scenes. In S. F. El-Hakim and A. Gruen, editors, *Videometrics and Optical Methods for 3D Shape Measurement, Electronic Imaging*, volume 4309, pages 107–116. SPIE International Society for Optical Engineering, 2001.

[20] J. O'Rourke. Uniqueness of orthogonal connect-the-dots. In G. T. Toussaint, editor, *Computational Morphology*, pages 97–104. Elsevier, 1988.

[21] D. Rappaport. On the complexity of computing orthogonal polygons from a set of points. Technical Report SOCS-86.9, McGill University, Montréal, Canada, 1986.

[22] R. C. Veltkamp. *Closed Object Boundaries from Scattered Points*, volume 885 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[23] C. Zhu, G. Sundaram, J. Snoeyink, and J. S. B. Mitchell. Generating random polygons with given vertices. *Computational Geometry: Theory and Applications*, 6:277–290, 1996.