On Sequential Triangulations of Simple Polygons

Robin Flatland [*]

Abstract

A triangulation is said to be Hamiltonian if its dual graph contains a Hamiltonian path. A sequential triangulation is a Hamiltonian triangulation having the additional property that the "turns" in the Hamiltonian path alternate left/right. Such triangulations are useful in computer graphics rendering and are related to a new type of two-guard walk. In this paper we present a simple $O(n \log n)$ algorithm that determines all sequential triangulations (or equivalently all sequential two-guard walks) of a simple n vertex input polygon. The previous best algorithm uses the polygon's visibility graph and hence runs in worse case $O(n^2)$ time [1].

1 Introduction

A triangulation is said to be *Hamiltonian* [1] if its dual graph contains a Hamiltonian path. A *sequential triangulation* is a Hamiltonian triangulation having the additional property that the "turns" in the Hamiltonian path alternate left/right. Specifically, the Hamiltonian path in the dual graph corresponds to a walk through the triangulation such that each step moves from one triangle to an adjacent triangle. For each triangle, the walk enters it through one edge and exits through one of the remaining two edges. The walk is said to take either a left or right turn, depending on which edge it crosses when it exits, as shown in Figure 1. The series of turns on the path must alternate between left and right for the triangulation to be sequential. Figure 2 is an example.

Sequential triangulations are useful in computer graphics rendering systems which often face a memory bus bandwidth bottleneck in the processor-to-graphics pipeline. Sequential triangulations help alleviate this problem because they have a succinct and simple encoding which reduces the amount of data transferred over the bus. See [1] for details. Sequential triangulations also have a relationship to a new type of two-guard walk which we call a *sequential walk*. In a *two-guard walk* [3] [4] of a simple polygon P, two guards start at a point s on P's boundary and travel along the boundary in opposite directions until they meet at a point t, while at all times being visible to each other. The walk is sequential if s and t are both vertices and the guards' motion is restricted to taking turns moving forward one vertex at a time towards t. We observe that every sequential walk has a corresponding sequential triangulation, and vice versa. The diagonals of the triangulation are the lines of sight connecting pairs of vertices that the guards occupied simultaneously during the walk. (See Figure 2.)

[*]Department of Computer Science, Siena College, `flatland@siena.edu`

In this paper we present an $O(n \log n)$ algorithm that identifies all sequential triangulations (all sequential walks) of a simple n vertex input polygon. The previous best algorithm uses the polygon's visibility graph and hence runs in worse case $O(n^2)$ time [1]. Related work includes Arkin et al.'s $O(n^2)$ algorithm for testing if a simple polygon has a Hamiltonian triangulation [1], a problem which was later solved in $O(n \log n)$ time [2], and then optimally in $O(n)$ time [5]. The latter two algorithms actually identify all vertex pairs admitting a *discrete straight walk*, which is similar to a sequential walk except that on a guard's turn he may move forward one *or more* vertices. In [2], Narasimhan notes that a polygon has a discrete straight walk *iff* it has a Hamiltonian triangulation, and he gives an $O(n)$ algorithm for constructing a Hamiltonian triangulation given any vertex pair that admits such a walk. He also shows that a Hamiltonian triangulation can be made sequential by adding at most $n/2$ Steiner points. A related but harder problem is that of determining if a triangulated surface model can be decomposed into k sequential tristrips, where a sequential tristrip is a sequence of adjacent triangles with alternating left/right turns. This problem has been shown to be NP-complete [9], and heuristics have been developed [6] [7].

1.1 Notation

We describe our algorithm from the perspective of computing sequential walks. In a walk, the guard moving in the clockwise (counter-clockwise) direction is called the $left$ ($right$) guard. A vertex *admits* a sequential walk if there exists a sequential walk starting at that vertex. Each vertex admits at most two sequential walks. The *left sequential walk* (*right sequential walk*) begins with the left (right) guard making the first move. Because the guards alternately move forward one vertex at a time until they meet, the rest of the walk is fixed once the first move has been made. For a simple polygon, P, let $v_0, v_1, ..., v_{n-1}$ be its vertices in clockwise order. Addition and subtraction on subscripts of the vertices are modulo n. Note that there is a left (right) sequential walk from s to t iff the reverse of that walk from t to s is also sequential.

Two points x and y on P's boundary divide it into two chains. The first, $C_{ccw}(x, y)$, extends in the counter-clockwise direction from x to y, and the second, $C_{cw}(x, y)$ extends in the clockwise direction from x to y. An *open* chain is the chain minus its two endpoints. For reflex ver-

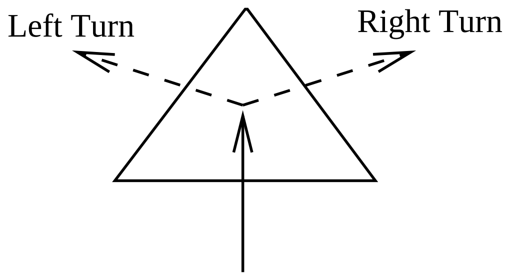

Figure 1: Left and right turn.

tex $v_i \in P$, the clockwise shot $S_{cw}(v_i)$ is the first point of P encountered by a "bullet" shot from v_i in the direction of the ray $\vec{v_{i-1}v_i}$. The shot $S_{ccw}(v_i)$ is defined similarly – see Figure 3. Wlog, we will assume the vertices are in general position so that no three are collinear.

2 Vertices Not Admitting Sequential Walks

Surrounding each reflex vertex of P is a contiguous interval of vertices that do not admit sequential walks. Consider for example vertex v_0 in Figure 3. Notice that the first half of the vertices on $C_{ccw}(v_0, S_{ccw}(v_0))$ (vertices v_0, v_{15}, v_{14}) admit neither a left nor a right sequential walk. Walks originating at these vertices result in the left guard "turning the corner" at v_0 before the right guard passes the point $S_{ccw}(v_0)$, thus causing the guards to lose sight of one another. Similarly, we can rule out left and right walks for the first half of the vertices on $C_{cw}(v_0, S_{cw}(v_0))$ (vertices v_0 and v_1). Because the number of vertices on $C_{cw}(v_0, S_{cw}(v_0))$ is odd, we can also rule out a right sequential walk starting at vertex v_2.

If a left (right) sequential walk starting from a vertex is ruled out, then the reverse of that walk can also be ruled out. For vertex v_0 in Figure 3, this means we can also rule out left and right sequential walks starting at vertices v_9, v_8, v_7, and v_6 (where v_1, v_0, v_{15}, and v_{14}'s sequential walks would have terminated). In addition, we can rule out a right sequential walk starting at vertex v_{10} (where v_2's right sequential walk would have terminated).

In general, for each reflex vertex v_i of P, we can rule out left and right walks for two contiguous intervals of vertices (one of which includes v_i) and up to four additional walks (either a left or a right walk) starting from vertices that are adjacent to these intervals. Given the two edges hit by v_i's shot points, the starting and ending vertex indices for the two intervals and the up to four additional vertices can easily be calculated in $O(1)$ time using modulo arithmetic. We will need to refer to the sequential walks that are **not** ruled out in this way. A vertex whose left (right) sequential walk is not ruled out will be called a L (R) vertex. A vertex can be both a L and a R vertex if neither walk is ruled out.

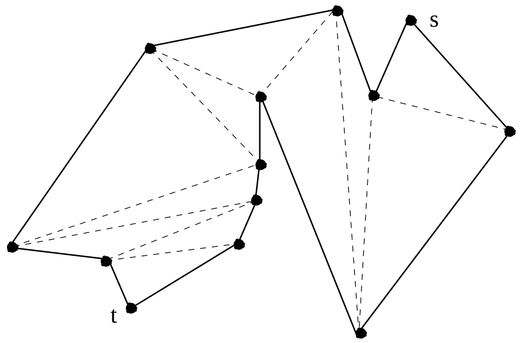

Figure 2: A sequential triangulation. Vertices s and t of the corresponding sequential walk are marked. Note the left guard makes the first move.

3 Algorithm

In the next section we will prove that every L (R) vertex admits a left (right) sequential walk. Once that is established, the algorithm for identifying the L and R vertices is simple: It first marks each vertex as both a L and R vertex. These marks will be selectively removed shortly. It then preprocesses the polygon in $O(n)$ time so that the first edge hit by a ray can be found in $O(\log n)$ time [8]. For each reflex vertex v_i, it determines the two edges hit by $S_{cw}(v_i)$ and $S_{ccw}(v_i)$ and calculates the starting and ending indices of the two intervals and the up to four additional vertices for which sequential walks can be ruled out. It removes the L or R mark from the up to four vertices not admitting one or the other sequential walk. The intervals, however, may overlap so it is not efficient to remove these L and R marks until all the intervals have been calculated. Once calculated, they can be consolidated in $O(n)$ time into a disjoint set of intervals covering the same range. This is done by first splitting intervals that wrap around from v_{n-1} to v_0 into two non-wrapping intervals, then sorting the intervals by their starting index (using bucket sort), and finally making a pass through them to consolidate them. The L and R marks of the vertices in the disjoint intervals are then removed. The vertices retaining a L or R mark admit sequential walks (sequential triangulations). The algorithm's running time is $O(n \log n)$, dominated by the time to compute the $O(n)$ shot points.

4 Proof that L (R) vertices admit sequential walks

It isn't difficult to see that starting a walk from a L or R vertex prevents the situation in which the guards lose sight of one another because one of them turns a corner at a reflex vertex (Figure 4a). But it is not apparent why some other part of the polygon could not obstruct their view (Figure 4b). Here we will show that neither situation can arise. Wlog, we will assume that n is even to simplify the presentation.

To show that all L (R) vertices admit left (right) sequential walks, we begin by establishing bounds on where one

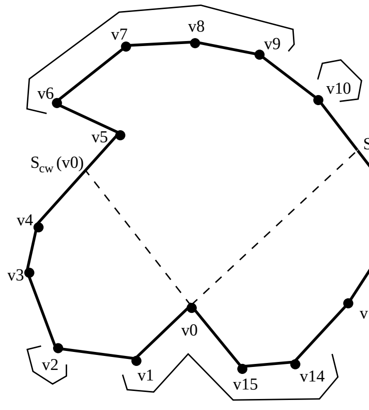

Figure 3: An example of the shot points $S_{ccw}(v_0)$ and $S_{cw}(v_0)$ for vertex v_0. The two intervals of vertices ruled out by v_0 are marked. Also marked are two additional vertices whose right sequential walks are ruled out by v_0.

guard is when the other is stationary at a reflex vertex during a *blind sequential walk* starting from a L or R vertex. A blind sequential walk is a sequential walk without the visibility requirement. Clearly every vertex admits two blind sequential walks.

Theorem 1 *Let s be a L (R) vertex. Consider two guards starting a blind sequential walk from s such that the left (right) guard makes the first move. For each reflex vertex $v_i \in P$, while one guard is traversing the edge $\overline{v_i v_{i+1}}$, the other guard will be stationary at a vertex on $C_{cw}(v_i, S_{ccw}(v_i))$. Similarly, while one guard is traversing edge $\overline{v_i v_{i-1}}$, the other guard is at a vertex on $C_{ccw}(v_i, S_{cw}(v_i))$.*

Proof. First we note that the same guard must traverse edges $\overline{v_i v_{i-1}}$ and $\overline{v_i v_{i+1}}$ since neither s nor t (both of which are L (R) vertices) can be a reflex vertex. Wlog, assume it is the left guard. If $s \in C_{ccw}(v_i, S_{ccw}(v_i))$, then clearly it must be on the second half of the chain. Thus the right guard will pass shot point $S_{ccw}(v_i)$ (and hence be on chain $C_{cw}(v_i, S_{ccw}(v_i))$) before the left guard turns the corner at v_i. If $s \notin C_{ccw}(v_i, S_{ccw}(v_i))$, then the entire path of the right guard lies in $C_{cw}(v_i, S_{ccw}(v_i))$. This establishes the first claim. The second claim can be shown similarly by considering the reverse of the walk. □

Theorem 2 *Let s be a L (R) vertex. Consider two guards starting a blind sequential walk from s such that the left (right) guard makes the first move. For each reflex vertex $v_i \in P$, while one guard is stationary at v_i, the other guard will traverse some part of chain $C_{cw}(S_{cw}(v_i), S_{ccw}(v_i))$.*

Proof. Wlog, say that it is the left guard that is stationary at v_i. Because the guards alternate moves, when the left guard is stationary at v_i, the right guard is traversing some edge $\overline{v_j v_{j-1}}$. Suppose for the sake of contradiction that no part of edge $\overline{v_j v_{j-1}}$ is on the chain $C_{cw}(S_{cw}(v_i), S_{ccw}(v_i))$. Then

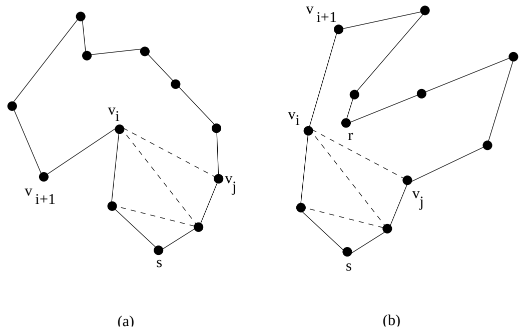

Figure 4: In both (a) and (b), the left guard is at v_i and the right guard is at v_j. It is the left guard's turn to move to v_{i+1}, but doing so results in the guards losing sight of one another. We show that neither situation can occur on blind sequential walks originating at a L or R vertex.

it either lies completely on open chain $C_{cw}(S_{ccw}(v_i), v_i)$ or $C_{cw}(v_i, S_{cw}(v_i))$. If it lies on $C_{cw}(S_{ccw}(v_i), v_i)$, then when the left guard next moves along edge $\overline{v_i v_{i+1}}$, the right guard will be stationary at v_{j-1} and $v_{j-1} \notin C_{cw}(v_i, S_{ccw}(v_i))$. This contradicts Theorem 1. A similar case can be made if the edge lies on $C_{cw}(v_i, S_{cw}(v_i))$ by noting that the right guard is at v_j when the left guard traverses $\overline{v_i v_{i-1}}$. □

Theorem 3 *If s is a L (R) vertex, then there is a left (right) sequential walk starting at s.*

Proof. Suppose for the sake of contradiction that s is a L (R) vertex, but there is not a left (right) sequential walk starting at s. Then if the two guards were to attempt a left (right) sequential walk from s, they would eventually reach vertices v_i and v_j such that either it is the left guard's turn to move but v_{i+1} and v_j are not visible to each other, or it is the right guard's turn to move but v_{j-1} and v_i are not visible. Consider the first situation in which it is the left guard's turn to move. There are two possible scenarios that might cause v_{i+1} and v_j to not be visible – either v_i is a reflex vertex such that the interior angle $\angle v_{i+1} v_i v_j$ is greater than 180 degrees, or some part (or parts) of chain $C_{cw}(v_{i+1}, v_j)$ cuts through the line of sight between v_{i+1} and v_j. See Figure 4.

The first scenario contradicts Theorem 1. In the second scenario, consider a line passing through v_{i+1} and v_j. Using a parallel sweep, move this line down until it reaches the last vertex on the part(s) of $C_{cw}(v_{i+1}, v_j)$ that cuts through the line of sight. Let r be that vertex, and note that it must be a reflex vertex. (See Figure 4b.) Notice that shot points $S_{cw}(r)$ and $S_{ccw}(r)$ are on the open chain $C_{ccw}(v_{i+1}, v_j)$. By theorem 2, if this walk were continued as a blind sequential walk, when one guard eventually reaches vertex r, the other guard would traverse some part of the chain $C_{cw}(S_{cw}(r), S_{ccw}(r))$. But here this is impossible since $C_{cw}(S_{cw}(r), S_{ccw}(r)) \subset C_{ccw}(v_{i+1}, v_j)$, which has already been traversed. □

5 Conclusions

Here we provide a new $O(n \log n)$ algorithm that determines all sequential triangulations of a simple input polygon. The algorithm is simple and asymptotically faster than previous algorithms. It is still an open question if this problem can be solved in $O(n)$ time.

References

[1] E. M. Arkin, M. Held, J. S. B. Mitchell, and S. Skiena. Hamiltonian triangulations for fast rendering. The Visual Computer **12**, pages 429–444. 1996.

[2] G. Narasimhan. On Hamiltonian Triangulations in Simple Polygons. Intl. J. of Comp. Geom. & Appls. **9:3**, pages 261–275. 1999.

[3] C. Icking and R. Klein. The Two Guards Problem. Intl. J. of Comp. Geom. & Appls. **2:3**, pages 257–285. 1992.

[4] L. H. Tseng, P. Heffernan, and D. T. Lee. Two Guard Walkability of Simple Polygons. Intl. J. of Comp. Geom. & Appls. **8:1**, pages 85–116. 1998.

[5] B. Bhattacharya, A. Mukhopadhyay, and G. Narasimhan. Optimal Algorithms for Two-Guard Walkability of Simple Polygons. LNCS **2125**, pages 438–449. 2001.

[6] F. Evans, S. Skiena, and A. Varshney. Optimizing Triangle Strips for Fast Rendering. In *IEEE Conf. on Visualization*, pages 319–326. 1996.

[7] X. Xiang, M. Held, and J. S. B. Mitchell. Fast and Effective Stripification of Polygonal Surface Models. In *Symp. on Interactive 3D Graphics*, pages 71–78. 1999.

[8] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. Algorithmica **2**, pages 209–233. 1987.

[9] R. Estkowski, J. S. B. Mitchell, and X. Xiang. Optimal Decomposition of Polygonal Models into Triangle Strips. In *Proc. ACM Symp. on Comp. Geom.*, pages 254-263. 2002.