International Journal of Computational Geometry & Applications

ON HAMILTONIAN TRIANGULATIONS IN SIMPLE POLYGONS [*]

GIRI NARASIMHAN [†]
Department of Mathematical Sciences,
The University of Memphis,
Memphis, TN 38152, USA.

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

An n-vertex simple polygon P is said to have a *Hamiltonian Triangulation* if it has a triangulation whose dual graph contains a hamiltonian path. Such triangulations are useful in fast rendering engines in Computer Graphics. We give a new characterization of polygons with hamiltonian triangulations and use it to devise $O(n \log n)$-time algorithms to recognize such polygons. We also give efficient algorithms for several related problems.

1. Introduction

A simple polygon P is said to have a *Hamiltonian Triangulation* if it has a triangulation whose dual graph contains a hamiltonian path. Such triangulations are useful in fast rendering engines in Computer Graphics, since most visualization of surfaces is currently done via triangles [1,4]. In order to transmit triangulation data for a simple polygon to a rendering engine, one needs to send information about each of its triangular faces. For a polygon with a hamiltonian triangulation, its triangulation data can be sent at a faster rate since consecutive triangles share a face, and hence only incremental changes need to be specified per triangle. Benefits include potential reduction in rendering time by a factor of three, and compression for storing models. *OpenGL*, a triangular-mesh rendering product from Silicon Graphics uses such a specification to render hamiltonian triangulations. In a related work, Evans et al. [4] report a program for generating triangular strips from polygonal models. Since interactive display rates are becoming increasingly important in virtual reality and visualization applications, many rendering engines use special-purpose hardware to speed up the rendering process by maintaining a buffer to store k previously transmitted vertices. For polygons with hamiltonian triangulations, the

[*]A preliminary version of this paper appeared in the Proceedings of the 5th International Workshop, WADS'97, Halifax, Nova Scotia, Canada, August 1997, 321–330.

[†]E-mail: giri@msci.memphis.edu. Supported in part by NSF Grant CCR-940-9752 and by a grant from Cadence Design Systems.

size of the buffer needed is minimum since only the last two transmitted vertices need to be stored.

Hence it would be useful to recognize such polygons, especially since the recognition need be done only once, while the rendering may be performed many times. The first step towards the recognition of polygons with hamiltonian triangulations is a characterization of such polygons. Arkin et al. [1] observed that a polygon has a hamiltonian triangulation if and only if it is *discretely straight walkable*.

They introduced the concept of *Discrete Straight Walkability*, which is a discrete version of the *Straight Walkability* concept introduced by Icking and Klein [7]. The original *walkability* concepts grew out of research on the *two-guards problem*, which in turn arose from the study of different versions of *art-gallery problems* (for example, see [8]). Briefly, the two-guards problem involves determining whether two guards can move along the boundary of a simple polygonal room from a common starting point s to a common destination point t while staying mutually visible at all times. Straight walkability deals with a version of the two-guards problem where the guards do not backtrack during their walk. Discrete straight walkability deals with a variant of straight walkability where only one guard moves at any one time while the other guard is stationary at a vertex. Guard problems are important because they contribute to the study of visibility in polygons. It is interesting to note its relationship to triangulations in polygons as well.

Using the connections to discrete straight walkability, Arkin et al. [1] showed an algorithm to recognize polygons with hamiltonian triangulations in time that is linear in the size of the visibility graph of the given polygon P. The size of the visibility graph of P could be quadratic in the size of P and hence their algorithm could be very inefficient even for nearly convex polygons.

In section 4 we show two new characterizations of discretely straight walkable polygons. Using these characterizations we show more efficient algorithms for many problems related to such polygons. In particular we show (in section 5) the following results:

1. There exists an $O(n \log n)$-time algorithm to recognize simple polygons with a hamiltonian triangulation, i.e., simple polygons that are discretely straight walkable.

2. Given a simple polygon and two vertices s and t with respect to which the polygon is discretely straight walkable, there exists an $O(n)$-time algorithm to construct a discrete straight walk or a hamiltonian triangulation of the polygon.

3. Given vertices s and t on a simple polygon, there exists an $O(n)$-time algorithm to determine whether the polygon is discretely straight walkable with respect to the two vertices.

4. There exists an $O(n \log n)$-time algorithm to list out all pairs of points on a simple polygon with respect to which the polygon is discretely straight walkable.

The algorithms in [1] for the second and third problems had a time complexity of $O(|E|)$, where E is the set of edges in a visibility graph of the polygon. Our results improve on their time complexities. The other two problems have never been studied before. The last result subsumes the third result quoted above. The reason two of the four algorithms run in $O(n \log n)$ time instead of linear time is that the algorithms compute all the ray shots obtained by extending the edges. There does not seem to be any way of avoiding the computation of some of the ray shots. For example, in [5], the computation of redundant ray shots are avoided.

As observed in [1], even if a triangulation is hamiltonian, the topology of the triangulation is not necessarily specified by an encoding sequence of vertices. A variant of the concept of hamiltonian triangulations called *Sequential Triangulations* was introduced in [1]. It turns out that even in *OpenGL*, sequential triangulations can be more efficiently specified than hamiltonian triangulations. We show that any hamiltonian triangulation can be converted into a sequential triangulation by the introduction of at most $n/2$ steiner points.

2. Preliminaries

We define notation for this paper; much of our notation is borrowed from [7,5]. A *polygonal chain* in the plane is a concatenation of line segments. The endpoints of the segments are called *vertices*, and the segments themselves are *edges*. If the segments intersect only at the endpoints of adjacent segments, then the chain is *simple*, and if a polygonal chain is closed we call it a *polygon*. In this paper, we deal only with simple polygons. Two points in the polygon are (mutually) *visible* if the line joining them is contained in the polygon. We assume that the input is in general position, which means that no three vertices are collinear, and no three lines defined by edges intersect in a common point. If x and y are points of P, then $P_{CW}(x, y)$ ($P_{CCW}(x, y)$) is the subchain obtained by traversing P clockwise (counterclockwise) from x to y.

In most of the problems we consider, two vertices of P are specially designated as the start vertex, s, and the goal vertex, t. The subchains $P_{CW}(s, t)$ and $P_{CCW}(s, t)$ are denoted by L and R, respectively, and will be sometimes referred to as the left side chain and the right side chain respectively. By definition, both L and R are oriented from s to t. Points on L (R) are denoted by p, p', p_1, etc. (q, q', q_1, etc.). If p is a point (not necessarily a vertex) of a polygonal chain, we denote by $Succ(p)$ the vertex of the chain immediately succeeding p, and by $Pred(p)$ the vertex preceding p. For $p, p' \in L$, we say that p *precedes* p' (and p' *succeeds* p) if we encounter p before p' when traversing L from s to t. We write $p < p'$.

A *(general) walk* is a pair of continuous functions

$$f : [0, 1] \rightarrow L, \quad g : [0, 1] \rightarrow R,$$

where $f(0) = g(0) = s$, $f(1) = g(1) = t$, and $f(t)$ and $g(t)$ are co-visible for all t. A walk is *straight* if f and g are monotonic functions; that is, $f(t_1) \leq f(t_2)$ and $g(t_1) \leq g(t_2)$ whenever $t_1 < t_2$. A straight walk is a *discrete straight walk* if the following is true for all values $0 \leq x < y \leq 1$: if $f(x)$ and $f(y)$ are distinct points on the same edge (except its endpoints), then $g(x) = g(y)$ and is equal to some vertex on R, and vice versa. A point $q \in R$ is a *walk partner* for a point $p \in L$ (and vice versa), if there exists a walk with the two guards at points p and q respectively at some time during the walk. Depending on the context, we use the term walk partner for straight walks and discrete straight walks as well.

For points x and y, $d(x, y)$ is the direction of the directed segment from x to y. The ray $\vec{r}(x, y)$ is the ray with terminus x in direction $d(x, y)$. The line containing x and y is denoted by $\ell(x, y)$. An important definition is that of *(ray) shots*: the *backward (ray) shot* (or *hit point*) from a reflex vertex $p \in L$, denoted $Backw(p)$, is the first point of P encountered by a "bullet" shot from p in direction $d(Succ(p), p)$, and the *forward (ray) shot* $Forw(p)$ is the point encountered by the bullet shot from p in direction $d(Pred(p), p)$. We let $d(Backw(p))$ represent the direction of the backward shot from p, i.e. $d(Succ(p), p)$. We define the backward shot $Backw(q)$ and forward shot $Forw(q)$ for a reflex vertex $q \in R$ in similar fashion.

In some of the problems we consider in this paper, the points s and t may not be specified, in which case the chains L and R are not defined. In this situation, instead of the $Succ$ and $Pred$, functions, we define the functions $CSucc$ and $CPred$, respectively. For a point p on the polygon P, $CSucc(p)$ is defined as the vertex that (strictly) follows p in a clockwise traversal (of P) starting from p; $CPred(p)$ is defined as the vertex that (strictly) follows p in a counterclockwise traversal (of P) starting from p. Also, instead of backward (forward, respectively) ray shot from a reflex vertex, we define clockwise (counterclockwise, respectively) ray shots. The clockwise (counterclockwise, respectively) ray shot from a reflex vertex $p \in P$, denoted by $Clockw(p)$ ($CClockw(p)$, respectively), is the first point of P encountered by a "bullet" shot from p in direction $d(CPred(p), p)$ ($d(CSucc(p), p)$, respectively). The endpoint of the ray shot is denoted by $rcw(p)$ ($rccw(p)$, respectively).

The *visibility graph* of the vertices of a simple polygon is the graph obtained by representing each of the vertices of the polygon by vertices in the graph and having an edge between two vertices in the graph if the corresponding two vertices in the polygon are visible.

3. Previous Work: Straight Walkability

In order to explain our results, we refer to the two-guards problem studied by Icking and Klein [7]: Given a simple polygon, P with n vertices and two distinguished vertices s and t, determine if two guards can move from s to t along the two boundary chains while remaining mutually visible at all times. If such a movement is possible then the polygon P is said to be *walkable* with respect to s and t. Furthermore, if it is possible for the two guards to reach the destination t without ever backtracking along the way, then the polygon P is said to be *straight walkable* with respect to s and t. Additionally, if it is possible for the two guards to move from s to t in such a way that when one guard is moving the other guard is stationary at a vertex, then the polygon is said to be *discretely straight walkable* with respect to s and t. We will simply use the terms walkable polygon, straight walkable polygon, or discretely straight walkable polygon to mean that there exists a pair of vertices on the polygon with respect to which it is walkable, straight walkable, or discretely straight walkable, respectively.

A simple polygon is said to be LR-visible with respect to two points on its boundary if the two chains determined by the two points are weakly visible from each other, i.e., every point on one chain is visible from some point on the other. As before, we will simply use the term LR-visible polygon to refer to a simple polygon that is LR-visible with respect to some two points on the polygon.

The main result of Icking and Klein's paper [7] deals with a precise characterization of straight walkable polygons. They consider two types of forbidden configurations, known as *deadlocks* and *wedges*, which are shown in Figs. 1(a) and (b). Each of the cases is shown consisting of the two polygonal chains L and R and the ray shots going from one chain to the other causing the forbidden configuration. The first two cases (Fig. 1(a)) show a deadlock with respect to s and t respectively. Here, the clockwise portion of the polygon between a and b is referred to as a deadlock. It is a forbidden configuration if exactly one of s or t lies inside the deadlock. This is because it would be impossible for the two guards to go past the points a and b without losing sight of each other. The next two cases (Fig. 1(b)) show a wedge configuration. Here, the clockwise portion of the polygon between a and b is referred to as a wedge. It is a forbidden configuration if it lies on chain L or on chain R, i.e., both s and t lie outside it. This is because it would be impossible for a guard to traverse the two bends in the wedge portion without losing sight of the other guard. Note that in all the four configurations the two ray shots (dotted) must intersect in the interior of the polygon). As explained above, an instance of each of the configurations prevents a polygon from being straight walkable; [7] show that the absence of these configurations is also sufficient to ensure straight walkability of a LR-visible polygon (see theorem 1 below).

Theorem 1 [7] *A simple polygon P is* straight walkable *with respect to two of its vertices s and t if and only if:*

1. *P is LR-visible with respect to s and t,*
2. *P does not have any* deadlocks *with respect to either s or t, and*
3. *P does not have any* wedges *on the two boundary chains L and R.*

In the process of developing an algorithm to recognize straight walkable polygons, Icking and Klein [7] developed a considerable collection of definitions, functions and theorems, some of which are summarized here since these are necessary for our work. They define functions lo and hi on the vertices of P. Specifically, the following definitions are given for L, with symmetric definitions applying to R. (The operations min and max are defined with respect to the ordering on the chain L, so that min of a set of points is a point of the set not preceded by any other.)

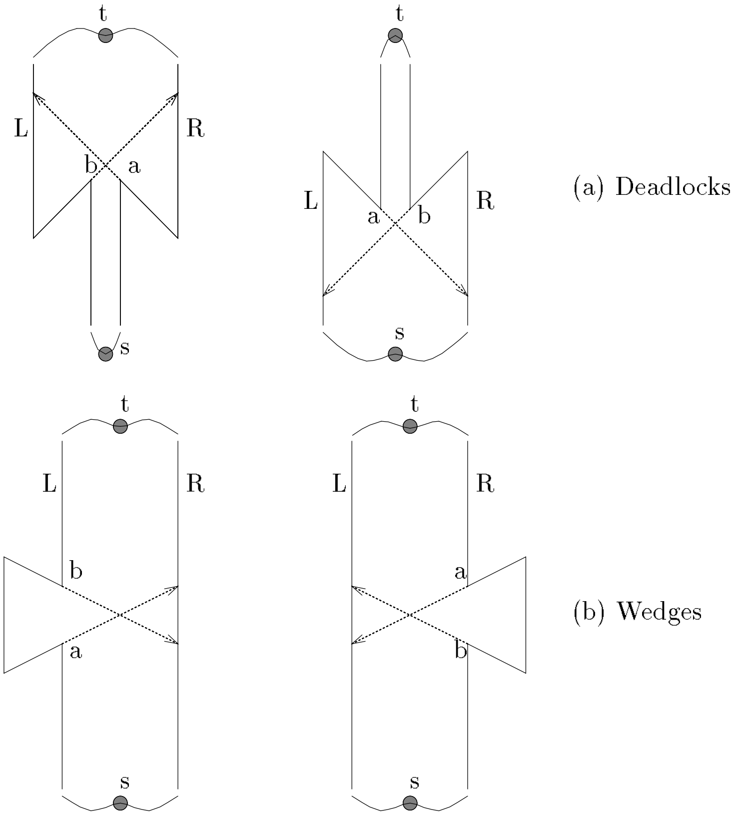

Figure 1: Forbidden configurations for straight walkable polygons

Definition 1 [7] *For a vertex $p \in L$, define:*

$$
\begin{aligned}
hiP(p) &= \min\{q \mid q \text{ vertex of } R \text{ and } L \ni Backw(q) > p\} \\
hiS(p) &= \min\{Forw(p') \in R \mid p' \text{ vertex of } L_{>p}\} \\
hi(p) &= \min\{hiP(p), hiS(p), t\} \\
loP(p) &= \max\{q \mid q \text{ vertex of } R \text{ and } L \ni Forw(q) < p\} \\
loS(p) &= \max\{Backw(p') \in R \mid p' \text{ vertex of } L_{<p}\} \\
lo(p) &= \max\{loP(p), loS(p), s\}
\end{aligned}
$$

We can think of hi and lo as functions from the vertices of L to the points of R, and also from the vertices of R to the points of L. Icking and Klein [7] point out that these are monotonic functions and they prove the following theorem.

Theorem 2 [7] *A simple polygon P is* straight walkable *with respect to two of its vertices s and t if and only if $[lo(v), hi(v)]$ is a non-empty interval for every vertex $v \in P$.*

Furthermore, they also showed that if P is straight walkable, then the set of possible *walk partners* for a vertex v is precisely the points of the interval $[lo(v), hi(v)]$,

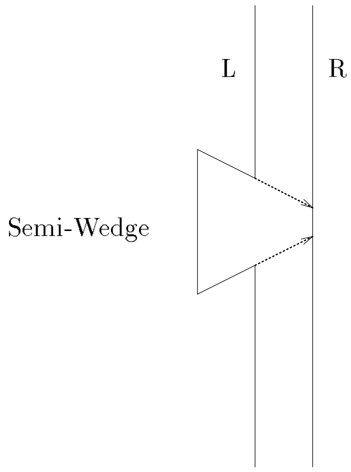

Figure 2: Forbidden configuration for discretely straight walkable polygons

where two points $p \in L$, $q \in R$ are walk partners in a given straight walk if the two guards are at points p and q at some moment of the walk.

4. Characterization

Theorems 3 and 4 give precise characterizations of discretely straight walkable polygons, i.e., polygons with hamiltonian triangulations. For the characterization, we consider a variant of the *wedge* configuration, which we call as the *semi-wedge* configuration. Fig. 2 shows a semi-wedge formed on the chain L. A similar one for R is not shown. Again, note that the two ray shots (dotted) must either intersect in the interior of the polygon or they must terminate in the interior of the same edge of the polygon. We show that this is a forbidden configuration for discretely straight walkable polygons. We go on to show that this condition is also sufficient for a straight walkable polygon to be discretely straight walkable.

Hence a stricter version of the wedge condition is enough to ensure discrete straight walkability, as proved in theorem 4 below. However, we first prove the following characterization (theorem 3) of discretely straight walkable polygons. Compare this characterization to theorem 2 in [7].

Theorem 3 *A simple polygon P is discretely straight walkable if and only if, for each vertex $v \in P$, the interval $[lo(v), hi(v)]$ contains a vertex of the polygon.*

Proof. Assume that the polygon P is discretely straight walkable. Since every discretely straight walkable polygon is straight walkable, for each vertex $v \in P$, the interval $[lo(v), hi(v)]$ is non-empty. If, for some two points s and t, there is a straight walk from s to t, then every vertex v on the left (right, resp.) side chain has a walk partner (for a straight walk) on the right (left, resp.) side chain. Furthermore, if there is a discrete straight walk from s to t, then every vertex v on the left (right, resp.) side chain has a walk partner (for a discrete straight walk) that is a vertex of the right (left, resp.) side chain. Thus, for each vertex $v \in P$, the interval $[lo(v), hi(v)]$ contains some vertex of the polygon.

To prove the converse, assume that for every vertex $v \in P$, the interval $[lo(v), hi(v)]$ contains some vertex of the polygon. For the sake of contradiction, assume that the polygon is not discretely straight walkable. This can only happen if for some adjacent vertices v_i, v_{i+1} on L (the argument is symmetric for R), $[lo(v_i), hi(v_i)] \cap [lo(v_{i+1}), hi(v_{i+1})]$ is devoid of vertices, since in this case, the two guards cannot complete the discrete straight walk.

We define the following two functions on the vertices of L and R:

$$\begin{aligned} loV(v) &= \min\{q \mid q \text{ vertex from } [lo(v), hi(v)]\} \\ hiV(v) &= \max\{q \mid q \text{ vertex from } [lo(v), hi(v)]\} \end{aligned}$$

Note that the values of loV and hiV must exist for both the vertices v_i and v_{i+1} by our assumption. Furthermore, since lo and hi are monotonic functions, so are loV and hiV. It was also shown in [7] that $x \in [lo(y), hi(y)]$ if and only if $y \in [lo(x), hi(x)]$. We now argue that if $hiV(v_i) = x_j$ and $loV(v_{i+1}) = x_k$, then $k = j+1$, since otherwise there would be a vertex between x_j and x_k, say x_l whose interval $[lo(x_l), hi(x_l)]$ would be devoid of vertices. Since every vertex is visible from its interval $[lo(v), hi(v)]$, we infer the following facts: v_i is visible from x_j, v_{i+1} is visible from x_{j+1}. Since v_i must be visible from v_{i+1} just as x_j is from x_{j+1}, either v_i or v_{i+1} must be visible from the entire edge joining x_j and x_{j+1}. This contradicts the assumption for the values of either $hiV(v_i)$ or $loV(v_{i+1})$. $\square$

The proof of the above theorem gives us a way of constructing the discrete straight walk (and consequently the hamiltonian triangulation), if one exists. This is because the vertices in the interval $[lo(v), hi(v)]$ provide a set of possible walk partners for the vertex v. The construction is described in the next section.

Theorem 4 *A simple polygon P is* discretely straight walkable *with respect to two of its vertices s and t if and only if:*

1. *P is straight walkable with respect to s and t,*
2. *P does not have any* semi-wedges *on the two boundary chains L and R.*

Proof. It is clear that an instance of a semi-wedge prevents a polygon from being discretely straight walkable, since a vertex inside the semi-wedge would not have a walk partner for a discrete straight walk.

Next we prove sufficiency of the two conditions for a polygon to be discretely straight walkable. Assume that the polygon is not discretely straight walkable. Thus, by theorem 3, we know that for some vertex v, the interval $[lo(v), hi(v)]$ is empty, or, for some vertex v, the interval $[lo(v), hi(v)]$ is non-empty, but devoid of vertices. If, for some vertex v, the interval $[lo(v), hi(v)]$ is empty, then the polygon is not straight walkable, and the first of the two conditions is violated. If, on the other hand, for some vertex v, the interval $[lo(v), hi(v)]$ is non-empty, but devoid of vertices, then this implies that $lo(v)$ and $hi(v)$ are not vertices either, and that they lie on a common edge of the polygon. Furthermore, this implies $lo(v) = loS(v)$ and

$hi(v) = hiS(v)$, since loP and hiP take only vertex values. But this is precisely a semi-wedge configuration, and hence the contradiction. □

5. Algorithmic Applications

The previous section gives us a characterization for discretely straight walkable polygons. In this section we present efficient algorithms for several variants of the discrete straight walkability problem. We first consider the problem of testing whether a given simple polygon has a hamiltonian triangulation with respect to two given points, i.e., is the polygon discretely straight walkable with respect to the two given points. Next we describe how to find a hamiltonian triangulation (or a discrete straight walk) of a simple polygon with respect to two given points. Before considering two other variants of the hamiltonian triangulation problem, in section 5.3, we present a $O(n \log n)$-time algorithm to enumerate all semi-wedges of a simple polygon. Using this algorithm, we show $O(n \log n)$-time algorithms for recognizing polygons with hamiltonian triangulations, i.e., polygons that are discretely straight walkable, and for computing all pairs of points between which a given polygon has a discrete straight walk.

5.1. Walkability with respect to two given points

Given a simple polygon P, and two points s and t on it, Heffernan [5] presents a surprising linear-time algorithm for finding hi and lo values by ignoring redundant ray shots and computing only relevant information. The search structure used in his algorithm makes it more efficient. We use Heffernan's algorithm and compute the hi and lo values for each vertex in linear time. By theorem 3, the only work left to check for discrete straight walkability with respect to the given points s and t is to check whether any of the output intervals is devoid of vertices. Clearly this can be achieved in optimal linear time.

5.2. Constructing a hamiltonian triangulation

Given a simple polygon P, and two points s and t on it, we showed above how to check in linear time if a discrete straight walk exists from s to t. We now show how to actually construct the discrete straight walk and the corresponding hamiltonian triangulation of P. The strategy for doing this is as follows. Theorem 3 shows that if P is discretely straight walkable (i.e., it has a hamiltonian triangulation), then loV and hiV values exist for each vertex of the chains L and R and that the intervals $[loV(v), hiV(v)]$ intersect for adjacent vertices on each of the two chains. Now the simple strategy for constructing the discrete straight walk is going to be: the guards move alternately (with guard one moving first) in such a way that when the stationary guard is at vertex v, the other guard moves until the vertex $hiV(v)$ is reached.

It is now easy to generate the hamiltonian triangulation. Note that only one guard moves at a time; while this guard moves along an edge from one vertex to another, the other guard remains stationary during that period at a vertex. In terms of visibility, this means that the vertex holding the stationary guard is visible from the entire edge along which the mobile guard moved. This defines a triangle of the hamiltonian triangulation. Thus, every time the mobile guard moves to the next vertex, the edge connecting the stationary guard to the new position of the mobile guard is an edge of the triangulation.

It also gives the next triangle visited by a hamiltonian path in the dual of the triangulation. Thus the sequence of triangles generated by this method defines the hamiltonian path in the dual of the triangulation.

Clearly, this algorithm runs in optimal linear time, once the points s and t are known.

5.3. Computing all semi-wedges

Our algorithm for computing all semi-wedges of a simple polygon is similar to the algorithm by Tseng et al. [9] for computing all wedges of a simple polygon. The modifications are described below.

The algorithm involves first computing the endpoints of the clockwise and counterclockwise ray shots from every reflex vertex in the polygon. For a reflex vertex x, recall that these endpoints are denoted by $rcw(x)$ and $rccw(x)$ respectively. Using an algorithm due to Hershberger and Suri [6], the endpoint of any single ray shot can be computed in $O(\log n)$ time after a preprocessing step that takes $O(n \log n)$ time and $O(n)$ space. Thus all the $O(n)$ endpoints of the ray shots can be computed in $O(n \log n)$ time.

The vertices of the polygon along with the endpoints of the ray shots are then sorted according to the order in which they appear in a clockwise traversal of the input polygon. This sorted list L is stored as a circular linked list. Along with each item x in the list we also store information indicating whether it corresponds to a reflex vertex of the polygon, non-reflex vertex of the polygon, endpoint of a clockwise ray shot, or endpoint of a counterclockwise ray shot. If x is a reflex vertex, we store pointers to the endpoints of the clockwise and counterclockwise ray shots ($rcw(x)$ and $rccw(x)$, respectively) from the vertex. If x is the endpoint of a ray shot, we store a pointer to the corresponding reflex vertex (called $reflex(x)$) from which the ray shot originated. In this case, we store yet another pointer (called $nextv(x)$) to the reflex vertex of the polygon that appears next in the sorted list. The items of the list are given integer indices to facilitate the tests described below. Building the above data structure can clearly be performed in $O(n \log n)$ time.

Next we describe how to check, given two reflex vertices x and y, whether the clockwise portion of the polygon from x to y, $P_{CW}(x, y)$, forms a semi-wedge. This can be verified by first checking whether the points x, y, $rcw(y)$, $rccw(x)$ appear in that order in the list L, and secondly whether $P_{CW}(rcw(y), rccw(x))$ has at least one vertex (reflex or non-reflex) of the polygon. Checking whether the points appear in that order in L can be done in $O(1)$ time by inspecting the integral indices of these items and taking into account the possible circular order. Checking whether there is a vertex of the polygon between the items $rcw(y)$ and $rccw(x)$ can be done by checking whether the points $rcw(y)$, $nextv(rcw(y))$, and $rccw(x)$ appear in that order in the list L. Once again, this can be done in $O(1)$ time by inspecting the indices of these three items.

As mentioned above, we need an algorithm that will output all semi-wedges of a given simple polygon. A semi-wedge is denoted by the two reflex vertices that are its endpoints. The existence of a semi-wedge implies that for a discrete straight walk, one of the endpoints must lie in the semi-wedge. Since, semi-wedges that completely contain another semi-wedge do not serve any additional purpose, we describe below an algorithm that will output all "minimal" semi-wedges (i.e., do not contain other semi-wedges). This algorithm works as follows: for each reflex vertex x of the polygon, it will output the "minimal" semi-wedge that has x as its counterclockwise endpoint. We are thus left to describe how to find a minimal semi-wedge with x as its counterclockwise endpoint.

We traverse clockwise from x and for each reflex vertex y that is encountered we test whether $P_{CW}(x, y)$ forms a semi-wedge (using the scheme described above). We only need the first reflex vertex y for which the clockwise portion of the polygon from x to y forms a semi-wedge. For the next iteration, we move x to the next reflex vertex in the clockwise direction. One important observation is that the search for the corresponding reflex vertex y at the counterclockwise end of the semi-wedge can start where we left off in the previous iteration. This observation helps to show that the time complexity of this phase of the algorithm is $O(n)$, since both endpoints of all semi-wedges are output by scanning the list once with no backtracking.

Putting all the pieces together, all semi-wedges of the simple polygon can be output in the form of a set of intervals in $O(n \log n)$ time with $O(n)$ space.

5.4. Recognizing polygons with hamiltonian triangulations

We will, in fact, solve a more general problem. We will show how to compute all pairs of points with respect to which a given polygon is discretely straight walkable. Other problems of this type that have been considered before include the problems of computing all LR-visible pairs of points [2] and all weakly-visible chords [3] of a simple polygon, and the problem of computing all pairs of points with respect to which a simple polygon is straight walkable [9]. As in the references mentioned, the output will be in the form of pairs of intervals.

Tseng et al. [9] describe a simple algorithm that takes as input the list of all LR-visible pairs of intervals, the list of all deadlocks and the list of all wedges of the polygon and that outputs the list of all pairs of points (x, y) such that the polygon is straight walkable with respect to the pair (x, y). It is clear that by replacing the list of all wedges by the list of all semi-wedges, the same algorithm would output all pairs of points (x, y) such that the polygon is discrete straight walkable with respect to the pair (x, y).

The problem of recognizing polygons with hamiltonian triangulations is now trivially solved (with the same running time): a simple polygon admits a hamiltonian triangulation if the output of the algorithm described above is non-empty.

Like the algorithm in [9], the algorithm to output all pairs of points with respect to which the polygon is discretely straight walkable takes $O(n \log n)$ time (and $O(n)$

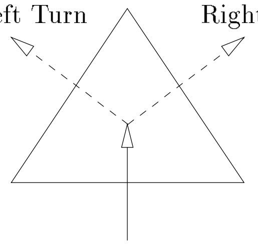

Figure 3: Left and Right Turns from a Triangle

space), as does the algorithm to recognize polygons with hamiltonian triangulations.

6. Sequential Triangulations and Steiner Points

The hamiltonian path in the dual of a hamiltonian triangulation can be imagined as a path that steps on the triangles. The path enters a triangle through one of its sides and leaves it through another side. Since there are two choices of sides to leave the triangle, the choices can be thought of as a left turn or a right turn, as shown in Fig. 3.

As defined in [1], a hamiltonian triangulation is called a *sequential triangulation* if the hamiltonian path in the dual of the triangulation involves a sequence of alternate left and right turns. Otherwise it is called a *non-sequential triangulation.* When transmitting information about hamiltonian triangulations to a rendering engine, one also needs to specify the turn direction for each turn, since otherwise ambiguities are possible [1]. Consequently, sequential triangulations are desirable since the turn directions are implicit and need not be transmitted. The interesting question of how to add steiner points such that a simple polygon permits a hamiltonian triangulation was considered in [1]. Evans et al. [4] showed that every polygon with more than four sides can be sequentially triangulated with an interior point. Here we show that any hamiltonian triangulation can be transformed into a sequential one by adding steiner points to selected triangles and then retriangulating them.

In order to understand the following discussion, we treat the hamiltonian path in the dual of the triangulation as a string of length n with symbols from the alphabet $\{L, R\}$, where L and R represent a left and right turn respectively. A sequential triangulation with n triangles is represented by one of two possible strings – $LRLR\ldots$ or $RLRL\ldots$. A non-sequential triangulation is represented by a string with the subsequence LL or RR in it.

Adding a steiner point to a triangle, retriangulating it, and then modifying the hamiltonian path to traverse all three triangles is equivalent to replacing the subsequence L (R, respectively) by the subsequence RLR (LRL, respectively). We refer to this transformation as applying transformation Π to that triangle. As shown in Fig. 4, the subsequence LL (two consecutive left turns) gets transformed into $LRLR$. In fact, the triangulation in Fig. 4(b) is a sequential one.

Let the hamiltonian path in the dual of the triangulation be a path through the

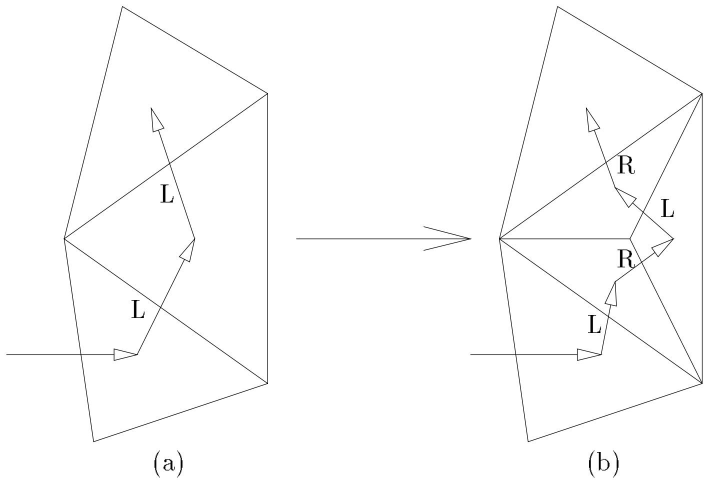

Figure 4: Adding a Steiner Point and Transformation Π

sequence of triangles $T_1, T_2, \ldots, T_n$, and let the turns be represented by the string $S = t_1, t_2, \ldots, t_n$, where $t_i \in \{L, R\}, 1 \leq i \leq n$. Thus t_i represents a turn in triangle T_i. Let $I = \{i_1, i_2, \ldots, i_r\}$ be the set of indices such that $t_j = t_{j+1}$, for $j \in I$, i.e., there are two consecutive left turns at indices j and $j+1$. Note that if there are 3 consecutive left turns, then there will be two entries in I corresponding to the first and the second left turn.

Let i and j be two consecutive indices in I. Consider S', the subsequence of S between i and j. Thus $t_i = t_{i+1}$, $t_j = t_{j+1}$, and $t_k \neq t_{k+1}, i < k < j$. It is easy to see that by applying transformation Π to the triangles $T_{i+1}, T_{i+2}, \ldots, T_j$ will change subsequence S' to a subsequence

$$t_{i'} t_{i'+1} \ldots t_{j'}$$

with no consecutive left or right turns. Also, $t_{i-1} \neq t_{i'}$ and $t_{j'} \neq t_{j+1}$.

Given the string of turns S, consider the two strings S_1 and S_2 defined as follows:

$$\begin{aligned} S_1 &= t_1 t_2 \ldots t_{i_1} t_{i_2+1} t_{i_2+2} \ldots t_{i_3} t_{i_4+1} t_{i_4+2} \ldots \\ S_2 &= t_{i_1+1} t_{i_1+2} \ldots t_{i_2} t_{i_3+1} t_{i_3+2} \ldots t_{i_4} t_{i_5+1} t_{i_5+2} \ldots \end{aligned}$$

Thus, S_1 and S_2 contain alternate portions of S between two consecutive left or right turns. Without loss of generality, assume that $|S_1| \leq |S_2|$. Thus $|S_1| \leq n/2$. Based on the observations made above, it is easy to see that adding a steiner point to each triangle corresponding to the turns in S_1 produces a sequential triangulation, thus proving the claim.

Consider the example shown in Fig. 5. The only hamiltonian path in the dual is one that involves all left turns or all right turns. Clearly, if the edges of the original triangulation are to be retained, then $n/2$ steiner points are necessary and sufficient. It is necessary since otherwise, there would be two original triangles next

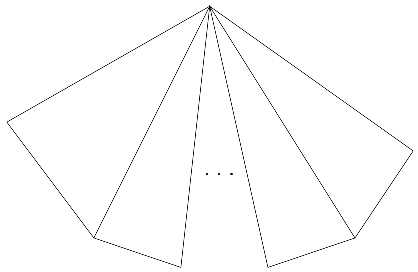

Figure 5: Worst Case Example for Adding Number of Steiner Points

to each other, which would involve two left (or right) turns. It is sufficient because adding a steiner point to every alternate triangle and performing the transformation mentioned above would convert it into a sequential triangulation.

We note that adding one steiner point increases the number of triangles (and consequently, the amount of data points transmitted for rendering) by two. Thus, adding $n/2$ steiner points could effectively double the number of data points transmitted for rendering, and may make it infeasible in practice. It is no cheaper than the cost of repeating a vertex, i.e., sending a degenerate triangle to simulate the LL or RR turn. While the main reason for preferring a sequential triangulation is because ambiguities may arise [1], these ambiguities may not always arise on a LL or RR turn (as also seen in the example in Fig. 5). In such cases, the addition of a steiner point may be avoided. As mentioned in [1], a better alternative to indicate a repeated left or right turn may be to transmit a swap flag or swap bit (if possible).

7. Open Problems

It is known [1] that given any set of points in the plane there exists a hamiltonian triangulation of these points and that such a triangulation can be computed efficiently. The most interesting related problem that remains open is the following [1]: given a set of points in 3-dimensional space, does there exist a *hamiltonian tetrahedralization* of the points? This problem is open even if the set of points are in convex, general position. Another variant of this problem is to consider hamiltonian triangulations in the skeleton of a tetrahedralization of the set of points. We can show that when the points are in convex, general position in 3-dimensional space, both the variants are equivalent. This problem seems to be related to variants of the *unfolding* problem for polytopes.

Other interesting problems relate to discrete versions of LR-visibility and their possible connections to discrete straight walkability.

References

1. E. M. Arkin, M. Held, J. S. B. Mitchell, and S. S. Skiena. Hamiltonian triangulations for fast rendering. *Visual Comput.*, 12(9):429–444, 1996.
2. G. Das, P. Heffernan, and G. Narasimhan. LR-visibility in polygons. *Comput.*

Geom. Theory Appl., 7:37–57, 1997.

3. G. Das, P. J. Heffernan, and G. Narasimhan. Finding all weakly-visible chords of a polygon in linear time. *Nordic J. Comput.*, 1:433–456, 1994.
4. F. Evans, S. S. Skiena, and A. Varshney. Optimizing triangle strips for fast rendering. *Proceedings of VISUALIZATION*, 1996.
5. P. J. Heffernan. An optimal algorithm for the two-guard problem. *Internat. J. Comput. Geom. Appl.*, 6:15–44, 1996.
6. J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.
7. C. Icking and R. Klein. The two guards problem. *Internat. J. Comput. Geom. Appl.*, 2(3):257–285, 1992.
8. J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.
9. L. H. Tseng, P. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *Internat. J. Comput. Geom. Appl.*, 8(1):85–116, 1998.