CONNECTING POLYGONIZATIONS VIA STRETCHES AND TWANGS

MIRELA DAMIAN [1], ROBIN FLATLAND [2], JOSEPH O'ROURKE [3],
AND SUNEETA RAMASWAMI [4]

[1] Dept. of Computer Science, Villanova Univ., Villanova, PA 19085, USA.
E-mail address: mirela.damian@villanova.edu

[2] Dept. of Computer Science, Siena College, Loudonville, NY 12211, USA.
E-mail address: flatland@siena.edu

[3] Dept. of Computer Science, Smith College, Northampton, MA 01063, USA.
E-mail address: orourke@cs.smith.edu

[4] Dept. of Computer Science, Rutgers University, Camden, NJ 08102, USA.
E-mail address: rsuneeta@camden.rutgers.edu

Abstract. We show that the space of polygonizations of a fixed planar point set S of n points is connected by $O(n^2)$ "moves" between simple polygons. Each move is composed of a sequence of atomic moves called "stretches" and "twangs," which walk between weakly simple "polygonal wraps" of S. These moves show promise to serve as a basis for generating random polygons.

1. Introduction

This paper[1] studies polygonizations of a fixed planar point set S of n points. Let the n points be labeled p_i, $i = 0, 1, \ldots, n-1$. A *polygonization* of S is a permutation σ of $\{0, 1, \ldots, n-1\}$ that determines a simple (non-self-intersecting) polygon $P = P_\sigma = (p_{\sigma(0)}, \ldots, p_{\sigma(n-1)})$. We will abbreviate "simple polygon" to *polygon* throughout. We do not make any general position assumptions about S, except to assume the points do not lie in one line so that there is at least one polygon whose vertex set is S. A point set S may induce as few as 1 polygon, if S is in convex position.[2] For a simple construction that induces $2^{\Theta(n)}$ polygons see Fig. 1a, and [CHUZ01] for additional details.

Our goal in this work is to develop a computationally natural and efficient method to explore all polygonizations of a fixed set S. One motivation is the generation of "random polygons" by first generating a random S and then selecting uniformly at random a polygonization of S. Generating random polygons efficiently is a long unsolved problem; only heuristics [AH96] or algorithms for special cases [ZSSM96], [HHH02] are known. Our work can be viewed as following a suggestion in [ZSSM96]:

2000 ACM Subject Classification: Nonnumerical Algorithms: F.2.2; Discrete Mathematics: G.2.
Key words and phrases: polygons, polygonization, random polygons, connected configuration space.

[1] Revision of earlier extended abstract [DFOR08].

[2] S is in convex position if every point in S is on the hull of S. and as many as 4.64^n polygons [GNT00].

> "start with a ... simple polygon and apply some simplicity-preserving, reversible operations ... with the property that any simple polygon is reachable by a sequence of operations"

Our two operations are called *stretch* and *twang* (defined in Sec. 2.2). Neither is simplicity preserving, but they are nearly so in that they produce polygonal wraps defined as follows.

Definition 1.1. *A* polygonal wrap $\mathcal{P}_\sigma$ *is a polygonal chain* $p_{\sigma(1)}, p_{\sigma(2)}, \ldots$, *determined by a sequence* σ *of point indices drawn from* $\{0, 1, \ldots, n-1\}$ *with the following properties:*

(1) *Every index in* $\{0, 1, \ldots, n-1\}$ *occurs in* σ.
(2) *Indices may be repeated. If index* i *appears more than once in* σ*, we call* p_i *a* point of multiple contact.
(3) *There exists an arbitrarily small perturbation of the points in multiple contact that makes the polygonal chain non-self-intersecting.*

Thus polygonal wraps disallow proper crossings[3] but permit self-touching. This notion is called a "weakly simple polygon" in the literature, but we choose to use our terminology to emphasize the underlying fixed point set and the nature of our twang operation. If a point p appears exactly twice in a polygonal wrap, we call p a *point of double contact*. Fig. 1b shows a polygonal wrap with five double-contacts (p_1, p_4, p_5, p_8 and p_9).

Stretches and twangs take one polygonal wrap to another. A stretch followed by a natural sequence of twangs, which we call a *cascade*, constitutes a *forward move*. Forward moves (described in Sec. 2.3) take a polygon to a polygon, i.e., they are simplicity preserving. Reverse moves will be introduced in Sec. 6. A *move* is either a forward or a reverse move. We call a stretch or twang an *atomic move* to distinguish it from the more complex forward and reverse moves.

Our main result is that the configuration space of polygonizations for a fixed S is connected by forward/reverse moves, each of which is composed of a number of stretches and twangs, and that the diameter of the space is $O(n^2)$ moves. We can bound the worst-case number of atomic moves constituting a particular forward/reverse move by the geometry of the point set. Experimental results on random point sets show that, in the practical situation that is one of our motivations, the bound is small, perhaps even constant. We have also established a quadratic lower bound on the worst-case number of atomic operations as a function of n. Establishing a combinatorial upper bound has so far proven elusive and is an open problem.

One can view our work as in the tradition of connecting discrete structures (e.g., triangulations, matchings) via local moves (e.g., edge flips, edge swaps) [BH08]. Our result is comparable to that in [vLS82], which shows connectivity of polygonizations in $O(n^3)$ edge-edge swap moves through intermediate self-crossing polygons, and to that in [HHH02], which establishes noncrossing connectivity within special classes of polygonizations. The main novelty of our work is that we avoid proper crossings but achieve connectivity via polygonal wraps.

We begin by defining pockets, which play a central role in our algorithms for polygonal transformations. Then in Sec. 2.1 we describe two natural operations that transform one polygon into another but fail to achieve connectivity of the configuration space of polygonizations, which motivates our definitions of stretches and twangs in Sec. 2.2. Following

[3]Two segments properly cross if they share a point x in the relative interior of both, and cross transversely at x.

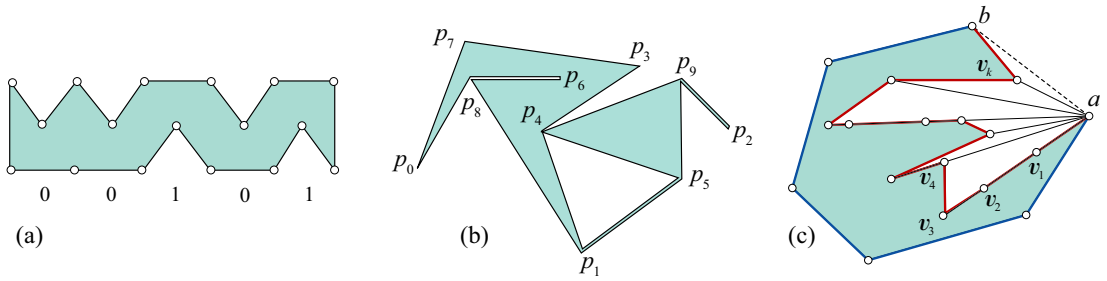

Figure 1: Examples. (a) A set of $n = 3k + 2$ points that admits 2^k polygonizations. (b) Polygonal wrap $\mathcal{P}_\sigma$ with $\sigma = (0, 8, 6, 8, 1, 5, 9, 2, 9, 4, 5, 1, 4, 3, 7)$ (c) A polygonization with one pocket with lid ab.

these preliminaries, we establish connectivity and compute the diameter in Secs. 3–7. We explore the possible application to random polygons in Sec. 8, establishing that a random walk through the polygonizations graph is ergodic, i.e., approaches a stable distribution that reaches all polygonizations. We conclude with open problems in Sec. 9.

1.1. Pockets and Canonical Polygonization

Let P be a polygonization of S. A *pocket lid* is an edge on the boundary of the convex hull of S that is not an edge of P. A *pocket* of P is a polygon external to P that is bounded by P and a pocket lid. For a fixed edge ab on the convex hull of S with a preceding b in counterclockwise order on the hull, we define the *canonical polygonization* of S to be a polygon with a single pocket with lid ab, in which the pocket vertices are ordered by angle about vertex a, and from closest to farthest from a if along the same line through a. We call this ordering the *canonical order* of the pocket vertices; see Fig. 1c. The existence of this canonical polygonization for any point set S not in convex position was established in [CHUZ01].

2. Polygonal Transformations

Let P be a polygon defined by a circular index sequence σ. We examine operations that permute this sequence, transforming P into a new polygon with the same set of vertices linked in a different order. Throughout the paper we use $\triangle abc$ to denote the closed triangle with corners a, b and c.

2.1. Local Transformations

The systematic study of constant-sized transformations that alter one simple polygon to another was initiated in [HHH02]. They defined a *k-flip* as an alteration of k (not necessarily consecutive) edges, and established a number of results, including showing that 3-flips are sufficient to connect polygonizations among several subclasses of polygons based on various visibility properties. But no constant k-flip move is known to be sufficient for connecting all simple polygonizations, and they conclude that "the connectivity of general simple polygons remains a challenging open problem." Although we do not resolve this open problem by a "local transformation" in their sense, we do resolve it by stepping outside their paradigm

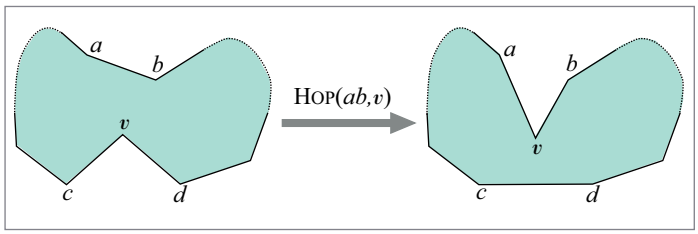

Figure 2: HOP(ab, v) illustrated.

in two regards: (1) We permit polygonal wraps as intermediate structures; and (2) Our atomic moves are local and constant-sized, but they cascade into sequences of as many as $\Omega(n^2)$ atomic moves.

The most natural local transformation is a swap transposition of two consecutive vertices of P that results in a new (non-self-intersecting) polygon. A swap is a particular 2-flip. Because this is easily seen as insufficient for polygonization connectivity, 3-flips were explored in [HHH02]. Much less obviously, even these were shown to be insufficient for connectivity, except within various polygon subclasses. We review one of their 3-flips, the "planar VE-flip," which we call a *hop*, because our *stretch* operation is a generalization of this.

The hop operation generalizes the swap by allowing a vertex to hop to any position in the permutation, as long as the resulting polygon is simple. Fig. 2 shows the stretching of the edge ab down to vertex v, effectively "hopping" v between a and b in the permutation. We denote this operation by HOP(e, v), where $e = ab$ (note the first argument is *from* and the second *to*).

To specify the conditions under which a hop operation is valid, we introduce some definitions, which will be used subsequently as well. A polygon P has two sides, the interior of P and the exterior of P. Let $abc = (a, b, c)$ be three noncollinear vertices consecutive in the polygonization P. We call vertex b a *true corner vertex* since the boundary of P takes a turn at b. We distinguish between the *convex side* of b, that side of P with angle $\angle abc$ smaller than π, and the *reflex side* of b, the side of P with angle $\angle abc$ larger than π. Note that this definition ignores which side is the interior and which side is the exterior of P, and so is unrelated to whether b is a convex or a reflex vertex in P. Every true corner vertex has a convex and a reflex side (collinear vertices will be discussed in Sec. 2.2). To ensure that the resulting polygon is simple, HOP(e, v) is valid if and only if the following two conditions hold: (1) the triangle induced by the two edges incident to v is empty of other polygon vertices and (2) the triangle induced by e and v lies on the reflex side of v and is empty of other polygon vertices.

Although more powerful than a swap, there also exist polygons that do not admit any hops, as was established in [HHH02], and so hops do not suffice to connect all polygonizations.

The limited transformation capabilities of these 2- and 3-flip operations motivate our introduction of two new operations, *stretch* and *twang*. The former operation relaxes the two hop conditions and allows the creation of a polygonal wrap. The latter operation restores the polygonal wrap to a polygon. We show that together they are capable of transforming any polygon into a canonical form (Secs. 3-5), and from there to any other polygon (Secs. 6-7).

2.2. Stretches and Twangs

Unlike the HOP(e, v) operation, which requires v to fully see the edge e into which it is hopping, the STRETCH(e, v) operation only requires that v see a point x in the interior[4] of e. The stretch is accomplished in two stages: (i) temporarily introduce two new "pseudovertices" on e in a small neighborhood of x (this is what we call STRETCH$_0$ below), and (ii) remove the pseudovertices immediately using twangs.

STRETCH$_0$. Let v see a point x in the interior of an edge e of P. By *see* we mean "clear visibility", i.e., the segment vx shares no points with ∂P other than v and x (see Fig. 3a). Note that every vertex v of P sees such an x (in fact, infinitely many x) on some e. Let x^- and x^+ be two points to either side of x on e, both in the interior of e, such that v can see both x^- and x^+. Two such points always exist in a neighborhood of x. We call these points *pseudovertices*. Let $e = ab$, with x^- on the same side of x as a. Then STRETCH$_0(e, v)$ alters the polygon to replace e with (a, x^-, v, x^+, b), effectively "stretching" e out to reach v by inserting a narrow triangle $\triangle x^- v x^+$ that sits on e (see Fig. 3b).

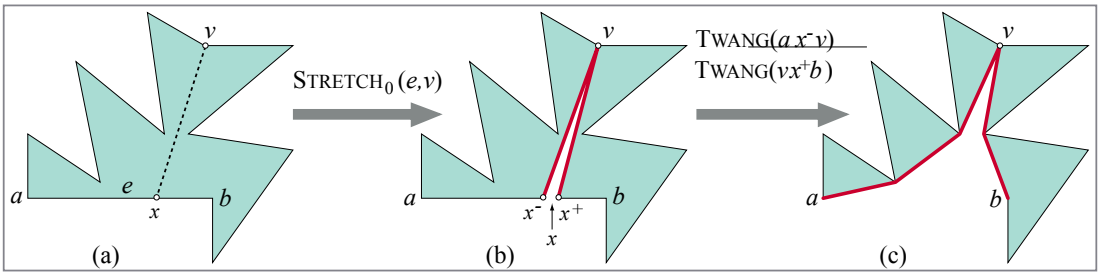

Figure 3: STRETCH(e, v) illustrated (a) v sees $x \in e$ (b) STRETCH$_0(e, v)$ (c) STRETCH(e, v).

To complete the definition of STRETCH(e, v), which removes the pseudovertices x^+ and x^-, we first define the twang operation.

TWANG. Informally, if one views the polygon boundary as an elastic band, a twang operation detaches the boundary from a vertex v and snaps it to v's convex side.

Definition 2.1. *The operation* TWANG(abc) *is defined for any three consecutive vertices* $abc \in \sigma$ *such that*

(1) $\{a, b, c\}$ *are not collinear.*
(2) b *is either a pseudovertex, or a vertex in multiple contact. If* b *is a vertex in multiple contact, then* $\triangle abc$ *does not contain a* nested *multiple contact at* b*. By this we mean the following: Infinitesimally perturb the vertices of* P *to separate each multiple contact, so that* P *becomes simple. Then* $\triangle abc$ *does not contain any other occurrence of* b *in* σ*. (E.g., in Fig. 4a,* $\triangle a'bc'$ *contains a second occurrence of* b *which prevents snapping* $a'bc'$ *to* b*'s convex side.)*

Under these conditions, the operation TWANG(abc) *replaces the sequence* abc *in* $\mathcal{P}$ *by* sp(abc)*, where* sp(abc) *indicates the shortest path from* a *to* c *that stays inside the closed* $\triangle abc$ *and does not cross* $\partial\mathcal{P}$*. We call* b *the* twang vertex*. Whenever* a *and* c *are irrelevant to the discussion, we denote the twang operation simply by* TWANG(b)*.*

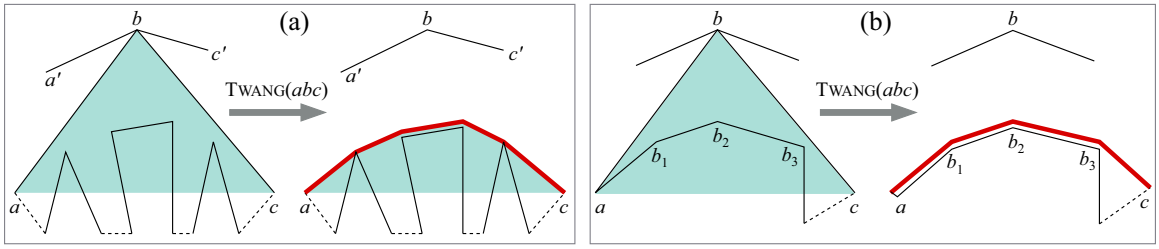

Figure 4: TWANG(abc) illustrated (a) TWANG(abc) replaces abc by sp(abc) (b) TWANG(abc) creates the hairpin vertex a and three doubled edges ab_1, b_1b_2 and b_2b_3.

Informally, TWANG(abc) "snaps" the boundary to wrap around the hull of the points in $\triangle abc$, excluding b (see Fig. 4a). A twang operation can be viewed as taking a step toward simplicity by either removing a pseudovertex or reducing the contact multiplicity of a vertex. We should note that sp(abc) includes every vertex along this path, even collinear vertices. If there are no points inside $\triangle abc$, then sp(abc) = ac, and TWANG(abc) can be viewed as the reverse of HOP(ac, b). If $a{=}c$ (i.e., ab and bc overlap in $\mathcal{P}$), we call b a *hairpin* vertex of $\mathcal{P}$; in this case, TWANG(aba) replaces aba in $\mathcal{P}$ by a. Hairpin vertices and "multiple edges" arise naturally from twangs. In Fig. 4b for instance, TWANG(abc) produces a hairpin vertex at a and doubled edges ab_1, b_1b_2, b_2b_3. So we must countenance such degeneracies. In general, there are points in the closed triangle, and the twang increases the contact multiplicity of some of these points. Below, we will apply twangs repeatedly to remove all multiple contacts.

STRETCH. We can now complete the definition of STRETCH(e, v), with $e = ab$. First execute STRETCH$_0$(e, v), which picks the two pseudovertices x^+ and x^-. Then execute TWANG(ax^-v) and TWANG(vx^+b), which detach the boundary from x^+ and x^- and return to a polygonal wrap of S (see Fig. 3c). We refer to e (v) as the *stretch edge* (*vertex*).

2.3. Twang Cascades

A twang in general removes one contact of the twang vertex and creates perhaps several others. A TWANGCASCADE applied on a polygonal wrap $\mathcal{P}$ removes all multiple contacts from $\mathcal{P}$. Note that for any point b of multiple contact, there always exists a vertex sequence abc that satisfies the twang conditions, and therefore the twang cascade loop never gets stuck. In general, there are several twang choices at any one step of the cascade. Although the selection order does not affect our proofs, there are cases where different orders will result in different final polygons at the end of a cascade. We therefore select a canonical ordering, always twanging the lowest-indexed point among the alternatives available.

[4]By "interior" we mean "relative interior," i.e., not an endpoint.

TwangCascade($\mathcal{P}$)

Loop for as long as $\mathcal{P}$ has a point of multiple contact b:

1. Among all the vertex sequences in $\mathcal{P}$ that satisfy the twang conditions (cf. Def. 2.1), select the lowest-indexed b in the sequence abc.
2. Twang(abc).

That a twang cascade eventually terminates is not immediate. The lemma below, shows that Twang(abc) shortens the perimeter of the polygonal wrap (because it replaces abc by $\mathrm{sp}(abc)$) by at least a constant δ depending on the geometry of the point set. Therefore, any twang cascade must terminate in a finite number of steps.

Lemma 2.2. *A single twang* Twang*(abc) decreases the perimeter of the polygonal wrap by at least $\delta = 2d_{\min}(1 - \sin(\alpha_{\max}/2))$, where $d_{\min}$ is the smallest pairwise point distance and $\alpha_{\max}$ is the maximum strictly convex angle formed by any triple of non-collinear points.*

Proof. Let $\mathcal{C}$ be the circle centered at b of radius $d_{\min}$, and let b_1 (b_2) be at the intersection of $\mathcal{C}$ and ab (bc) (see Fig. 5). Then $\mathrm{sp}(abc)$ is a convex path nested inside the convex quadrilateral ab_1b_2c. It follows from the theorem that, for two strictly nested convex bodies, the inner perimeter is less than the outer perimeter [BB04, p. 32], that $|\mathrm{sp}(abc)| < |ab_1| + |b_1b_2| + |b_2c|$. This in turn implies that the decrease in the perimeter is at least $|b_1b| + |bb_2| - |b_1b_2|$. Simple calculations show that $b_1b_2 = 2d_{\min}\sin(\alpha_{\max}/2)$, so the perimeter decreases by at least $\delta = 2d_{\min} - 2d_{\min}\sin(\alpha_{\max}/2)$, which concludes the proof. ∎

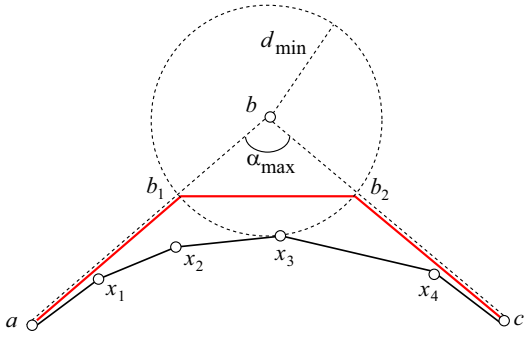

Figure 5: Lower bound on δ in the proof of Lemma 2.2.

Twang Cascade Bounds. We have been unsuccessful in obtaining a combinatorial upper bound on the number of twangs in any twang cascade. An impediment to establishing a bound is that the multiplicity of contacts at a point can decrease and then increase again in a twang cascade, as illustrated in Fig. 6. This example hints at the complex changes that can occur during a cascade, and why establishing an upper bound is problematical.

Figures 7 and 8 show that $\Omega(n)$ points can each twang $\Omega(n)$ times in one cascade, providing an $\Omega(n^2)$ worst-case lower bound on the length of a cascade. In Figure 7, the cascade is initiated by Stretch(e, v) followed by Twang(v). From there on Twang$(a_ib_ic_i)$ twangs in a cycle. Each such twang alters the path to $\mathrm{sp}(a_i, c_i)$, which wraps around b_{i+1}. In the next pass through the cycle, Twang$(a_ib_{i+1}c_i)$ occurs. This continues just twice in

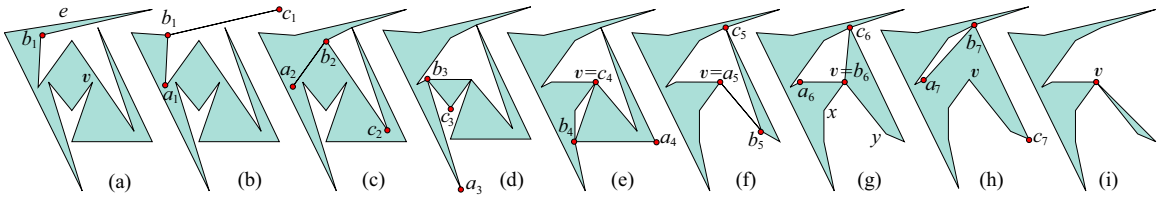

Figure 6: Point v twangs twice: (a) Initial P (b) After $\textsc{Stretch}(e, b_1)$ (c–h) After $\textsc{Twang}(a_i b_i c_i)$, $i = 1 \ldots 7$ (i) v in double contact a second time, and will twang a second time. If $\textsc{Twang}(xvy)$ were chosen in (g) instead of $\textsc{Twang}(a_6 v c_6)$, v would not twang twice.

Fig. 7, but in general the number of cycles is the number of b_j vertices inside each $\triangle a_i b_i c_i$. Fig. 8 shows that this number can be $\Omega(n)$. Here the b_j vertices are evenly spaced on a

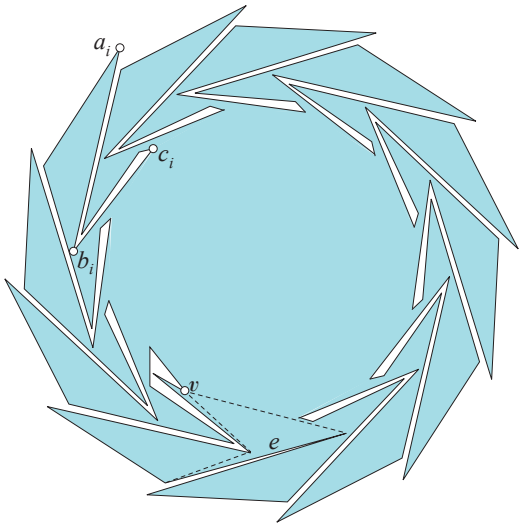

Figure 7: $\textsc{Stretch}(e, v)$ followed by $\textsc{Twang}(v)$ initiates a quadratic-length twang cascade.

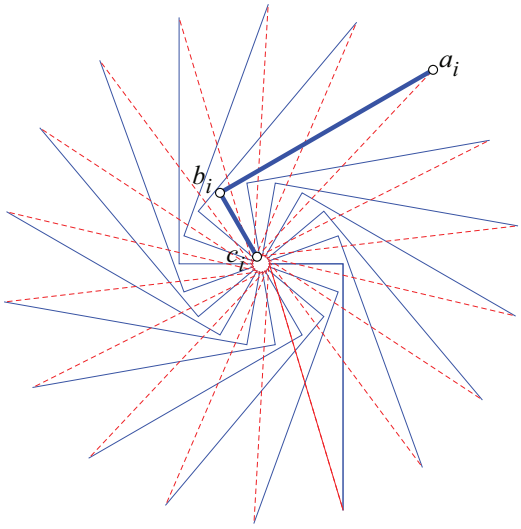

Figure 8: It is possible for each triangle $\triangle a_i b_i c_i$ to enclose $\Omega(n)$ vertices b_j.

circle, and $\angle(a_i, b_i, c_i) = 90°$. As the length $|a_i b_i|$ grows longer, the fraction of points inside $\triangle a_i b_i c_i$ approaches $n/4$.

Despite these weak bounds, our experiments suggest that the average length of a twang cascade for random S is about 1.2, with cascades of length > 7 rare.

Forward Move. We define a *forward move* on a polygonization P of a set S as a stretch (with the additional requirement that the pseudovertices on the stretch edge lie on the reflex side of the stretch vertex), followed by a twang and then a twang cascade, as described below:

ForwardMove(P, e, v)

Preconditions:
(i) P is a simple polygon
(ii) e and v satisfy the conditions of Stretch(e, v), and
(iii) v is a true corner vertex such that pseudovertices x^+ and x^- on e lie on the reflex side of v.
{Let u, v, w be the vertex sequence containing v in P (necessarily unique, since P is simple).}

1. $\mathcal{P} \leftarrow$ Stretch(e, v).
2. $\mathcal{P} \leftarrow$ Twang(uvw).
3. $P' \leftarrow$ TwangCascade$(\mathcal{P})$.

A ForwardMove takes one polygonization P to another P' (see Fig. 9), as follows from Lemma 2.2. Note that x^+ and x^- must lie on the reflex side of v (i.e., precondition (iii) of ForwardMove) so that Stretch(e, v) does not introduce a nested double contact in $\triangle uvw$ which would prevent the subsequent Twang(uvw). Next we discuss an important phenomenon that can occur during a forward move.

Stretch Vertex Placement. We note that the initial stretch that starts a move might be "undone" by cycling of the cascade. This phenomenon is illustrated in Fig. 9, where the initial Stretch(ab, v) inserts v between a and b in the polygonal wrap (Fig. 9b), but v ends up between c and b in the final polygonization (Fig. 9f). Thus any attempt to specifically place v in the polygonization sequence between two particular vertices might be canceled by the subsequent cascade. This phenomenon presents a challenge to reducing a polygon to canonical form (discussed in Sec. 5).

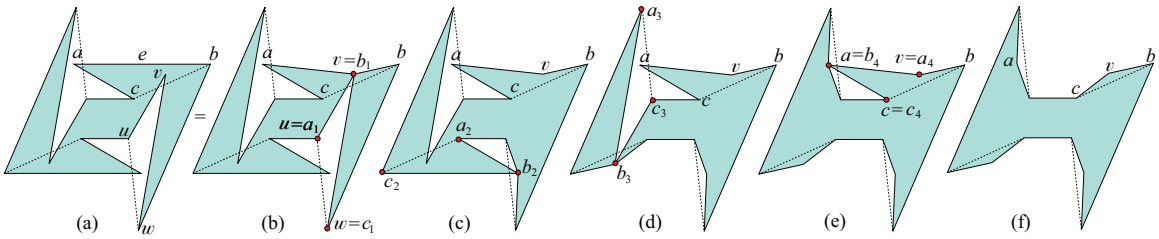

Figure 9: Forward move illustrated. (a) Initial polygon P (b) Step 1: Stretch(ab, v) (c) Step 2: Twang$(a_1b_1c_1)$ (d-f) Step 3: Twang$(a_2b_2c_2)$, Twang$(a_3b_3c_3)$, Twang$(a_4b_4c_4)$.

3. Single Pocket Reduction Algorithm

Now that the basic properties of the moves are established, we aim to show that our moves suffice to connect any two polygonizations of a point set S. The plan is to reduce an arbitrary polygonization to the canonical polygonization. *En route* to explaining this reduction algorithm, we show how to remove any particular pocket by redistributing its vertices to other pockets. This method will be applied repeatedly in Sec. 4 to move all pockets to one particular pocket.

In this section we assume that P has two or more pockets. We use $\mathcal{H}(P)$ to refer to the closed region defined by the convex hull of P, and $\partial\mathcal{H}(P)$ for its boundary. For a fixed hull edge ℓ that is the lid of a pocket A, the goal is to reduce A to ℓ by redistributing the vertices of A among the other pockets, using forward moves only. This is accomplished by the SINGLE POCKET REDUCTION algorithm, which repeatedly picks a hull vertex v of A and attaches v to a pocket other than A; see Fig. 10 for an example run.

SINGLE POCKET REDUCTION(P, ℓ) Algorithm

Loop for as long as the pocket A of P with lid ℓ contains three or more vertices:

1. Pick an edge-vertex pair (e, v) such that
   e is an edge of P on ∂B for some pocket $B \neq A$
   $v \in A$ is a non-lid true corner vertex on $\partial\mathcal{H}(A)$ that sees e
2. $P \leftarrow$ FORWARDMOVE(P, e, v).

We now establish that the SINGLE POCKET REDUCTION algorithm terminates in a finite number of iterations. First we prove a more general lemma showing that a twang operation can potentially reduce, but never expand, the hull of a pocket.

Lemma 3.1 (Hull Nesting under Twangs)**.** *Let A be a pocket of a polygonal wrap $\mathcal{P}$ and let vertex $b \notin \partial\mathcal{H}(\mathcal{P})$ satisfy the twang conditions. Let A' be the pocket with the same lid as A after* TWANG(b)*. Then $A' \subseteq \mathcal{H}(A)$.*

Proof. Let abc be the vertex sequence involved in the twang operation. Then TWANG(abc) replaces the path abc by $\mathrm{sp}(abc)$. If abc does not belong to ∂A, then TWANG(abc) does not affect A and therefore $A' \equiv A$. So assume that abc belongs to ∂A. This implies that b is a vertex of A. Note that b is a non-lid vertex, since $b \notin \partial\mathcal{H}(\mathcal{P})$. Then $\triangle abc \subset \mathcal{H}(A)$, and the claim follows from the fact that $\mathrm{sp}(abc) \subset \triangle abc$. ∎

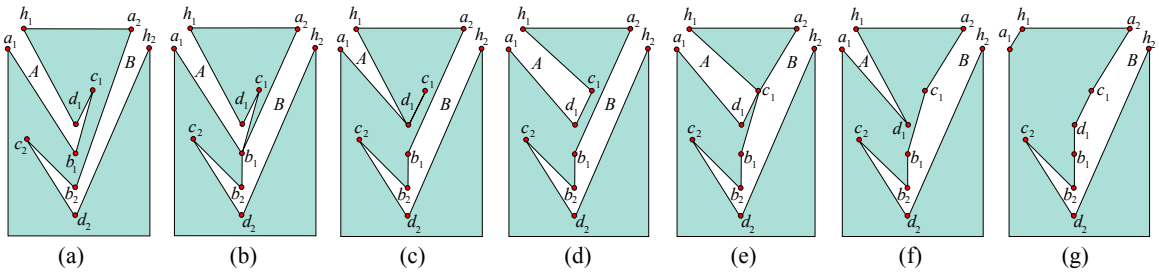

Figure 10: SINGLE POCKET REDUCTION(P, a_1h_1) illustrated: (a) Initial P; (b) After STRETCH(a_2b_2, b_1). (c) After TWANG$(a_1b_1c_1)$. (d) After TWANG$(c_1d_1h_1)$. (e) After STRETCH(b_1a_2, c_1). (f) After TWANG$(d_1c_1h_1)$. (g) After STRETCH(b_1c_1, d_1) + TWANG$(a_1d_1h_1)$.

Lemma 3.2. *The* SINGLE POCKET REDUCTION *algorithm terminates in $O(n)$ forward moves.*

Proof. Let S denote the set of vertices of P in $\mathcal{H}(A)$. Thus $|S| = O(n)$. We show that $|S|$ decreases by at least 1 in each loop iteration, thus establishing the claim of the lemma.

First observe that the existence of an edge vertex pair (e, v) selected in Step 1 is guaranteed by the fact that P has at least one pocket other than A. In particular, there is at

least one true corner vertex $v \in \partial\mathcal{H}(A)$ that sees points on the boundary of some pocket $B \neq A$ (otherwise, P has only one pocket). Step 2 of the SINGLE POCKET REDUCTION algorithm, which performs a forward move to a different polygonization, attempts to reduce A by vertex v, thus decrementing $|S|$. We now show that this step is successful in that it does not reattach v back to A. Furthermore, we show that S acquires no new vertices during this step. These together show that $|S|$ decreases by at least 1 in each loop iteration.

The first step of the forward move, STRETCH(e, v), does not affect S. The second step, TWANG(uvw), replaces the path uvw by sp(uvw), thus eliminating v from A. Because v is a true corner vertex of A, $\mathcal{H}(A)$ does not contain v at the end of this step. Let A' be the pocket of P with the same lid as A at the end of TWANGCASCADE(P). Since A's lid vertices never twang, Lemma 3.1 implies that $\mathcal{H}(A')$ is a subset of $\mathcal{H}(A)$ and therefore $|S|$ does not increase during the twang cascade. Furthermore, since $\mathcal{H}(A)$ does not contain v after the first twang operation, v must lie outside of $\mathcal{H}(A')$ at the end of the twang cascade. This concludes the proof. ∎

4. Multiple Pocket Reduction Algorithm

For a given hull edge e, the goal is to transform P to a polygon with a single pocket with lid e, using forward moves only. If e is an edge of the polygon, for the purpose of the algorithm discussed here we treat e as a (degenerate) target pocket T. We assume that, in addition to T, P has one or more other pockets, otherwise there is nothing to do. Then we can use the SINGLE POCKET REDUCTION algorithm to eliminate all pockets of P but T, as described in the POCKET REDUCTION algorithm below.

POCKET REDUCTION (P, e) Algorithm

If e is an edge of P, set $T \leftarrow e$, otherwise set $T \leftarrow$ the pocket with lid e
  (in either case, we treat T as a pocket).
For each pocket lid $e' \neq e$
  Call SINGLE POCKET REDUCTION(P, e')

Observe that the POCKET REDUCTION algorithm terminates in $O(n^2)$ forward moves: there are $O(n)$ pockets each of which gets reduced to its lid edge in $O(n)$ forward moves (cf. Lemma 3.2).

Fig. 11 illustrates the POCKET REDUCTION algorithm on a 17-vertex polygon with three pockets A, B and C, each of which has 3 non-lid vertices, and target pocket T with lid edge $e = t_1t_2$. The algorithm first calls SINGLE POCKET REDUCTION(P, a_1h_1), which transfers to B all non-lid vertices of A, so B ends up with 6 non-lid vertices (this reduction is illustrated in detail in Fig. 10). Similarly, SINGLE POCKET REDUCTION(P, a_2h_2) transfers to C all non-lid vertices of B, so C ends up with 9 non-lid vertices, and finally SINGLE POCKET REDUCTION(P, a_3h_3) transfers all these vertices to T. This example suggests that the bound $O(n^2)$ is, in fact, tight, as proved by the following lemma.

Lemma 4.1. *The* POCKET REDUCTION *algorithm employs* $\Theta(n^2)$ *forward moves, in the worst case.*

Proof. We have shown that the POCKET REDUCTION algorithm always terminates in $O(n^2)$ forward moves. We now show that there are cases in which the POCKET REDUCTION algorithm employs $\Theta(n^2)$ forward moves, thus proving the claim of the lemma.

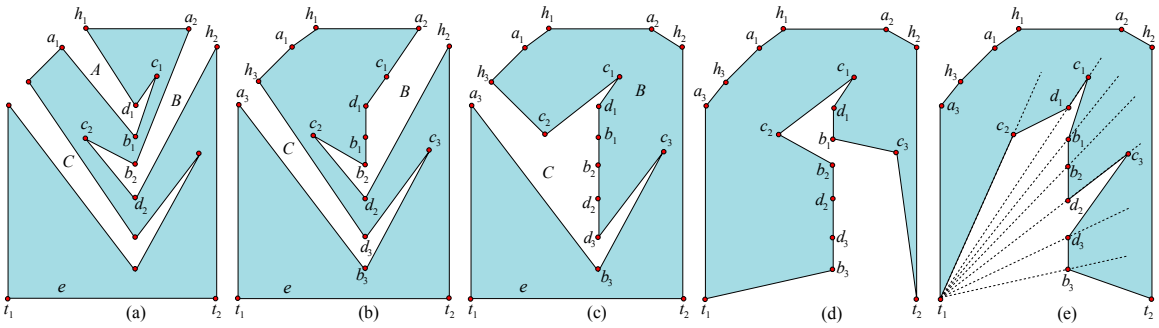

Figure 11: Pocket Reduction(P, t_1t_2): (a) Initial P. (b) After Single Pocket Reduction(P, a_1h_1). (c) After Single Pocket Reduction(P, a_2h_2). (d) After Single Pocket Reduction(P, a_3h_3). (e) After Canonical Polygonization(P, t_1t_2).

Consider an $(n = 5k)$-vertex polygon P with a structure similar to the one in Fig. 11a. Let A_i be the pocket of P with lid a_ih_i, for $i = 1, 2, \ldots, k$. Let $P_1 = P$ and

$$P_{i+1} \leftarrow \text{Single Pocket Reduction}(P_i, a_ih_i),$$

in which each forward move involves an edge of the pocket with lid $a_{i+1}h_{i+1}$. We will later show that such forward moves are always possible. We seek to prove that execution instances of Pocket Reduction(P) in which $P_2, P_3, \ldots, P_{k+1}$ are computed in this order, perform a total of

$$3\sum_{i=1}^{k} i = \Theta(n^2)$$

forward moves. Let A'_i be the pocket with lid a_ih_i in P_i. We prove by induction that the following invariants hold, for each i:

(1) A'_i contains all non-lid vertices of $A_1, A_2, \ldots, A_i$.
(2) For any $j > i$, A_j is the same in P_i and P (i.e., it does not get altered during the construction of $P_2, \ldots, P_i$).

An immediate consequence of invariants (1) and (2) is that the interior of $\mathcal{H}(A'_i)$ contains vertices of A'_i only, therefore the twang cascade initiated at a vertex of A'_i involves vertices of A'_i only (by Lem. 3.1). This implies that the number of forward moves employed in reducing A'_i is equal to the number of vertices of A'_i, which is equal to $3i$ (cf. invariant (1)), thus proving the claim on the total number of forward moves.

The base case corresponding to $i = 1$ is immediate, since $A'_1 = A_1$. To prove the inductive step, assume that invariants (1) and (2) hold for $P_1, P_2, \ldots, P_i$, for some $i < k$, and consider the polygon P_{i+1}. Invariant (2) tells us that A_{i+1} is identical in P_i and P. This along with invariant (1) implies that there exists true corner vertex $u \in \partial\mathcal{H}(A'_i)$ visible to an edge $e \in A_{i+1}$, meaning that ForwardMove(e, u) is possible. By choice, Single Pocket Reduction(P_i, a_ih_i) involves such a forward move. The result is that A_{i+1} absorbs u, $\partial\mathcal{H}(A'_i)$ loses u, and pockets A_j, for $j > i + 1$, remain unaltered. Identical arguments hold for subsequent forward moves involved in the reduction process for A'_i. At the end of this process, A'_{i+1} contains all vertices of A'_i, and the pockets A_j, for $j > i + 1$, are as in the original polygon P. This shows that the two invariants hold for A_{i+1}, thus completing the proof. ∎

5. Single Pocket to Canonical Polygonization

Let $P(e)$ denote an arbitrary one-pocket polygonization of S with pocket lid $e = ab$. Here we give an algorithm to transform $P(e)$ into the canonical polygonization $P_c(e)$. This, along with the algorithms discussed in Secs. 3 and 4, gives us a method to transform any polygonization of S into the canonical form $P_c(e)$. Our canonical polygonization algorithm incrementally arranges pocket vertices in canonical order (cf. Sec. 1.1) along the pocket boundary by applying a series of forward moves to $P(e)$.

CANONICAL POLYGONIZATION$(P, e = ab)$ ALGORITHM

Let $a = v_0, v_1, v_2, \ldots, v_k, v_{k+1} = b$ be the canonical order of the vertices of pocket $P(e)$.
For each $i = 1, 2, \ldots, k$

1. Set $\ell_i \leftarrow$ line passing through a and v_i
2. Set $e_{i-1} \leftarrow$ pocket edge $v_{i-1}v_j$, with $j > i - 1$
3. If e_{i-1} is not identical to $v_{i-1}v_i$, apply FORWARDMOVE(e_{i-1}, v_i).

We now show that the one-pocket polygonization resulting after the i-th iteration of the loop above has the points $v_0, \ldots, v_i$ in canonical order along the pocket boundary. (Note that this invariant ensures there is an edge (v_{i-1}, v_j) with $j > i-1$ in Step 2.) This, in turn, is established by showing that the FORWARDMOVE in the i-th iteration involves only points in the set $\{v_i, v_{i+1}, \ldots, v_k\}$. These observations are formalized in the following lemmas.

Lemma 5.1. *The i-th iteration of the* CANONICAL POLYGONIZATION *loop produces a polygonization of S with one pocket with lid e and with vertices $v_0, \ldots, v_i$ consecutive along the pocket boundary.*

Proof. The proof is by induction. The base case corresponds to $i = 1$ and is trivially true for the case when $e_0 = v_0v_1$. Otherwise, v_1 sees a subset of e_0 (since no edge can block visibility from v_0 to v_1) and therefore STRETCH(e_0, v_1) is possible. See Fig. 12a, where $e_0 = (v_0, v_3)$. Furthermore, v_1 may not twang a second time during the twang cascade of the forward move. This is because a second TWANG(v_1) may only be triggered by the twang of a hull vertex, which can never occur (hull vertices never twang). This implies that the FORWARDMOVE in Step 3 of the first iteration creates a one-pocket polygonization in which v_0 and v_1 are consecutive along the pocket boundary (see Fig. 12a,b). This completes the base case.

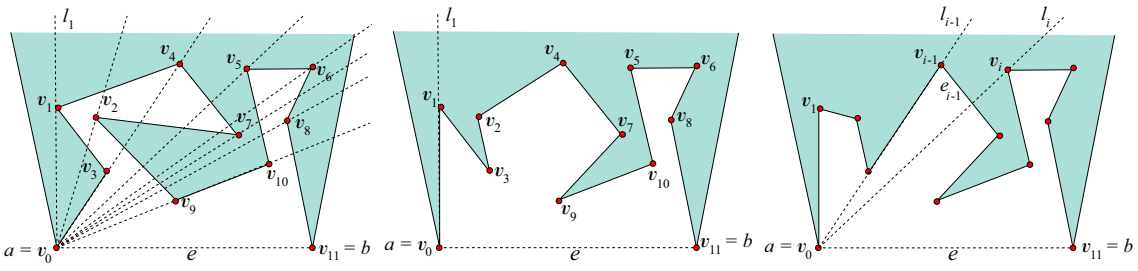

Figure 12: CANONICAL POLYGONIZATION: STRETCH(v_i, x) is always possible. (a) Base case ($i = 1$): v_1 sees v_0 (b) After iteration 1, v_0 and v_1 are consecutive along the pocket boundary (c) v_i sees e_{i-1}.

To prove the inductive step, suppose that the lemma holds for iterations $1, \ldots, i-1$. Note that the existence of the edge e_{i-1} selected in Step 2 of the algorithm follows immediately from the fact that $v_0, v_1, \ldots v_{i-1}$ are consecutive along the pocket boundary (by the inductive hypothesis). If e_{i-1} is identical to $v_{i-1}v_i$, there is nothing to prove. So assume that e_{i-1} and $v_{i-1}v_i$ are distinct. We now show that v_i sees e_{i-1}, so that $\textsc{Stretch}(e_{i-1}, v_i)$ is possible.

First observe that the wedge bounded by ℓ_{i-1} and ℓ_i is either degenerate (if v_0, v_{i-1}, v_i are collinear), or is empty of any pocket points (since v_i follows v_{i-1} in the clockwise sorted order). In the former case, v_i sees v_{i-1} and e_{i-1}. In the latter case, e_{i-1} must intersect ℓ_i (cf. Fig. 12c). In either case, v_i sees e_{i-1} and hence $\textsc{Stretch}(e_{i-1}, v_i)$ is possible. This along with the induction hypothesis implies that at the end of stretch operation, vertices $v_0, v_1, \ldots v_i$ are consecutive along the pocket boundary.

Next we show by contradiction that the twang cascade of the forward move involves only vertices $v_{i+1}, \ldots, v_k$, so that $v_0, v_1, \ldots v_i$ remain consecutive along the pocket boundary. Suppose the claim is false. For ease of presentation, define $\text{rank}(v_i) = i$. Let y be the first vertex with $\text{rank}(y) \leq i$ to get into double contact. Clearly y cannot coincide with a, since a is a hull vertex and cannot get into double contact. Let $\textsc{Twang}(qrs)$ be the twang that created the double contact at y. Note that at the time of $\textsc{Twang}(qrs)$, vertices $v_0, v_1, \ldots v_i$ are consecutive along the pocket boundary, since none of these vertices was in double contact prior to y (by choice of y) and therefore could not have twanged. Since $\textsc{Twang}(qrs)$ creates the double contact at y, $y \in \triangle qrs$ and lies on $\text{sp}(qrs)$. We also have that $\text{rank}(r) > i$, by our choice of y.

Two cases are possible: (i) y lies strictly to the left of ℓ_i, and (ii) y lies on ℓ_i. In either case, since $y \in \triangle qrs$ and r lies on or to the right of ℓ_i, it must be that $\min\{\text{rank}(q), \text{rank}(s)\} < \text{rank}(y)$. Suppose w.l.o.g that $\text{rank}(q) < \text{rank}(y)$. Thus we have that $\text{rank}(q) < \text{rank}(y) \leq \text{rank}(v_i) < \text{rank}(r)$. In other words $\text{rank}(r) - \text{rank}(q) \geq 2$, but since $q \in \{v_0, v_1, \ldots v_{i-1}\}$ and qr is an edge of the pocket, it must be that $\text{rank}(r) - \text{rank}(q) = 1$ (since $v_0, v_1, \ldots v_i$ are consecutive along the pocket boundary). Thus we have reached a contradiction. This completes the induction step. ∎

Lemma 5.2. *The* Canonical Polygonization *algorithm constructs $P_c(e)$ in $O(n)$ forward moves.*

Proof. The claim that the Canonical Polygonization algorithm constructs the canonical form $P_c(e)$ follows immediately from Lemma 5.1. The bound on the number of moves follows from the fact that once a point v_i is placed in the correct position in the pocket in the i-th iteration of the algorithm, its position will not be changed again. Hence, the number of forward moves required to place all points $v_1, \ldots, v_k$ in order on the pocket boundary is $O(k)$, which is $O(n)$. ∎

6. Reverse Moves

Connectivity of the space of polygonizations will follow by reducing two given polygonizations P_1 and P_2 to a common canonical form P_c, and then reversing the moves from P_c to P_2. Although we could just define a reverse move as a time-reversal of a forward move, it must be admitted that such reverse moves are less natural than their forward counterparts. So we concentrate on establishing that reverse moves can be achieved by a sequence of atomic stretches and twangs.

Reverse Stretch. The reverse of STRETCH(e, v) may be achieved by a sequence of one or more twangs, as illustrated in Fig. 13a. This result follows from the fact that the "funnel" created by the stretch is empty, and so the twangs reversing the stretch do not cascade.

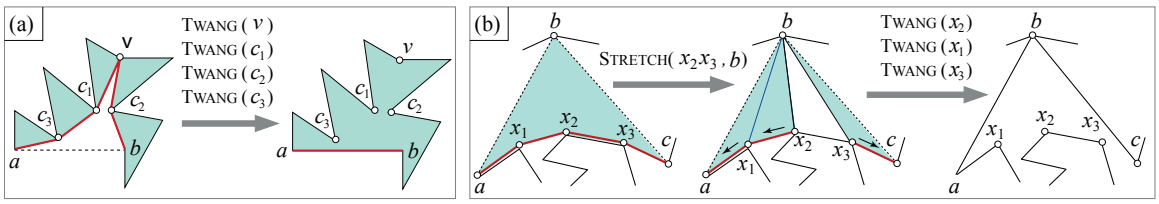

Figure 13: Reverse atomic moves: (a) STRETCH(ab, v) is reversed by TWANG(v), TWANG(c_1), TWANG(c_2), TWANG(c_3). (b) TWANG(b) is reversed by STRETCH(x_2x_3, b), TWANG(x_2), TWANG(x_1) and TWANG(x_3).

Reverse Twang. An "untwang" can be accomplished by one stretch followed by a series of twangs. Fig. 13b illustrates how TWANG(abc) may be reversed by one STRETCH(e, b), for any edge e of sp(abc), followed by zero or more twangs. Observe that the initial stretch in the reverse twang operation is not restricted to the reflex side of the stretch vertex, as it is in a FORWARDMOVE. If b is a hairpin vertex (i.e., a and c coincide), we view ac as an edge of length zero and the reverse of TWANG(b) is simply STRETCH(ac, b).

We have shown that the total effect of any forward move, consisting of one stretch and a twang cascade, can be reversed by a sequence of stretches and twangs. We call this sequence a *reverse move*. One way to view the consequence of the above two results can be expressed via regular expressions. Let the symbols s and t represent a STRETCH and TWANG respectively. Then a forward move can be represented by the expression st^+: a stretch followed by one or more twangs. A reverse stretch, s^{-1} can be achieved by one or more twangs: t^+. And a reverse twang t^{-1} can be achieved by st^*. Thus the reverse of the forward move st^+ is $(t^{-1})^+s^{-1} = (st^*)^+t^+$, a sequence of stretches and twangs, at least one of each.

7. Connectivity and Diameter of Polygonization Space

We begin with a summary of the algorithm which, given two polygonizations P_1 and P_2 of a fixed point set, transforms P_1 into P_2 using stretches and twangs only.

POLYGON TRANSFORMATION(P_1, P_2) Algorithm

1. Select an arbitrary edge e of $\partial\mathcal{H}(P_1)$.
2. $P_1 \leftarrow$ POCKET REDUCTION(P_1, e).
   $M_1 \leftarrow$ atomic moves of [$P_2 \leftarrow$ POCKET REDUCTION(P_2, e)].
3. $P_c \leftarrow$ CANONICAL POLYGONIZATION(P_1, e).
   $M_2 \leftarrow$ atomic moves of [CANONICAL POLYGONIZATION(P_2, e).]
4. Let $(M_1 \oplus M_2)^R$ be the reverse of the moves in $M_1 \oplus M_2$, where $\oplus$ represents concatenation).
5. For each stretch s (twang t) in $(M_1 \oplus M_2)^R$ in order,
   execute reverse stretch s^{-1}(reverse twang t^{-1}) on P_c.

This algorithm, along with Lemmas 4.1 and 5.2, establishes our main theorem:

Theorem 7.1. *The space of polygonizations of a fixed set of n points is connected via a sequence of forward and reverse moves. Each node of the space has degree in $O(n^2)$, and the diameter of the polygonization space is $O(n^2)$ moves.*

Proof. The first part of the theorem is established by the POLYGON TRANSFORMATION algorithm, whose correctness follows from Lemmas 4.1 and 5.2. The degree of a node is bounded by the number of vertex-edge pairs in the polygonization, which is $O(n^2)$ (see ahead to Fig. 14 for an example that achieves degree $\Omega(n^2)$). For any given polygonizations P_1 and P_2 of a fixed point set S, the POLYGON TRANSFORMATION(P_1, P_2) algorithm takes $O(n^2)$ forward/reverse moves (cf. Lemmas 4.1 and 5.2), so the diameter of the space of polygonizations of S is $O(n^2)$. ∎

This diameter bound is tight for our specific algorithm (cf. Lemma 4.1) but might not be for other algorithms. Each twang operation can be carried out in $O(n)$ time using a hull routine on the sorted points inside $\triangle abc$; and $\Omega(n)$ might be needed, because sp() might hit $O(n)$ vertices. So the running time of a single forward/reverse move is $T \cdot O(n)$, where T is an upper bound on the number of twangs in a move.

8. Random Polygon Generation

Define the *polygonization graph* G for a set of points S to have a node for each (simple) polygonization of S, and an arc connecting two polygonizations if they are connected by a single forward move. We define this graph as undirected, so it encompasses reverse moves as well. In this section we explore the possibility of using a random walk through G to generate "random" polygons. The random walk is a Markov chain that starts at some initial polygonization and then chooses among the available forward/reverse moves randomly according to some probability distribution. We will examine two different transition distributions below. We next address three fundamental questions: Is this Markov chain ergodic? If so, what is the stationary distribution? And does its mixing time qualify as "rapidly mixing"? For the purposes of random polygon generation, ideal answers would be: YES, UNIFORM, and YES. The answers we provide are: YES, UNIFORM OR NONUNIFORM, DEPENDING..., and LIKELY YES.

8.1. Ergodicity

A Markov chain is *ergodic* if all states are reached with positive probability, and if it is aperiodic [Ran06]. If the chain is ergodic, then the probability of any particular polygonization tends to a unique stationary distribution as the number of steps in the random walk $t \to \infty$. Ergodicity is clearly essential if the walk is to result in "random" polygons under any notion of randomness.

That all states are reached with positive probability is settled by connectedness (Theorem 7.1). In our context, aperiodicity reduces to the question of whether G is nonbipartite [Lov93, p. 356] (bipartite G can lead to periodic oscillations in probability).

Bipartiteness in turn depends on the choice of probability distribution for the random walk transitions. Perhaps the most natural choice is to select among the moves available from one polygonization with equal probability. Call this the *equal-transitions* model. We have indeed established that G is nonbipartite under this model, by showing that, for any set S (not in convex position), G contains a triangle. (Since G is undirected, there are closed

walks of length two; therefore, the existence of a closed walk of odd length is necessary and sufficient to ensure that the Markov chain is aperiodic.) We leave this nonbipartiteness as a claim, however, because an *unequal-transitions* model yields aperiodicity easily. If we add self-loops to each node of G by not transitioning with some positive probability, G is trivially nonbipartite. So, under either transitions model, the random walk through G is ergodic.

8.2. Stationary Distribution

It is well known that, for an ergodic Markov chain under the equal-transitions model, the stationary distribution assigns to each node p of G a probability proportional to the degree of p in G; more precisely, the probability is $\pi(p) = \deg(p)/(2E)$, where E is the number of edges of G [Lov93, p. 356]. Thus, the distribution is uniform only when G is regular. We now show with the example in Fig. 14 that G can be far from regular. The

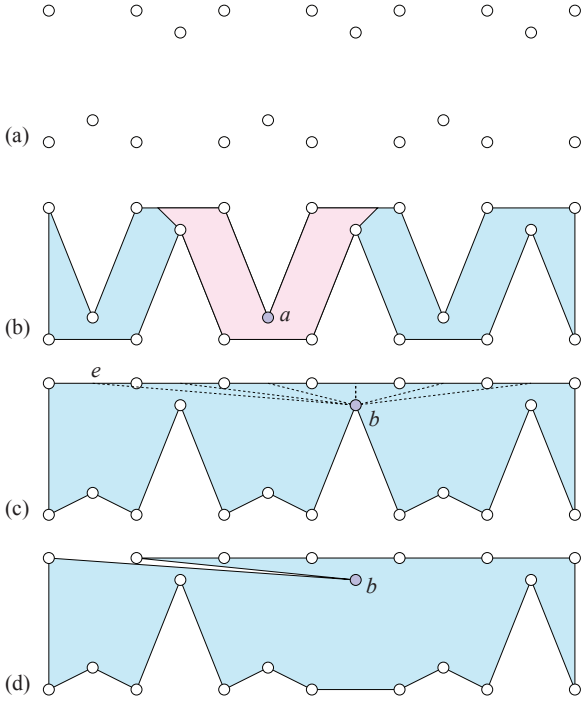

Figure 14: (a) Point set S. (b) A polygonization with degree $O(n)$. (c) A polygonization with degree $\Omega(n^2)$. (d) After Stretch(e, b) and Twang(b).

polygonization of S shown in (b) of the figure has the property that each vertex can see only a constant number of vertices and edges. For example, vertex a can see six vertices and seven edges. Because all moves are initiated by a stretch from an edge to a vertex, or vertex to a vertex (Sec. 6), this polygon has degree $O(n)$ in G. On the other hand, the polygonization shown in (c) has $\Omega(n)$ vertices each of which can see $\Omega(n)$ vertices and edges (e.g., b can see the entire top chain of the polygon). Each of the stretches determined by these (e, v) pairs leads after one twang to distinct polygons (e.g., (d)). Therefore this polygon node has degree $\Omega(n^2)$ in G.

Thus, not all polygonizations of S have the same degree in G, and therefore not all are equally likely destinations of a random walk through G under the equal-transitions model. In particular, the polygon in (c) has a higher probability than does the polygon in (b). This nonuniform distribution might be desired, for there is a sense in which it is natural that some polygonizations be more rarely reached than other "well-connected" polygonizations.

If, instead, a uniform distribution is desired, it can be achieved via an unequal-transitions model. We know that the maximum degree Δ of G is $\binom{n}{2}$. There is a well-known technique to alter the probabilities to achieve a uniform stationary distribution (see, e.g., [BH08] or [Sin93, p. 62]): from a node $p \in G$, move to an adjacent node with probability $1/\Delta$, and stay at p with probability $\deg(p)/\Delta$.

In our implementation, we followed the equal-transitions model so that the walk never remains at the same node.

8.3. Mixing Time

The mixing time of an ergodic Markov process is, roughly, the number of steps before the distribution π of polygonizations is close to the stationary distribution Π, i.e., it is the time needed for convergence. One can measure the distance between the distributions as the maximum of $|\Pi(p) - \pi(p)|$ over all nodes p of G. One definition of the mixing time [Ran06, Def. 3] (there are others) is expressed as a function of the closeness ε: $T(\varepsilon)$ is the minimum number of steps t so that the distance between the distributions is $\leq \varepsilon$ for all $t' > t$. A Markov chain is then called *rapidly mixing* if $T(\varepsilon)$ is bounded by a polynomial in $1/\varepsilon$ and n [Kan94], where for us n is the number of points in S. For example, it is well known that the mixing time for random walks on the n-dimensional hypercube graph is $O(n \log(n/\varepsilon))$ [Ran06, Thm. 3], and so these walks are rapidly mixing. Often the dependence on ε is suppressed; the mixing time for the hypercube is then $\Theta(n \log n)$.

Note that the configuration space G_H for the hypercube has $N = 2^n$ nodes, one for every n-bit binary number, but the mixing time is close to $\log N$. This logarithmic reduction from the size of the configuration space to the mixing time holds in a wide variety of circumstances,[5] and we can expect it in our situation. We mentioned earlier (Sec. 1) that the number of polygonizations N of a set of n points can be exponential. Indeed the maximum is known to be in $\Omega(4.6^n)$ [GNT00].[6]

Computing the mixing time is notoriously difficult (although see [MRS99] for a notable success). We were unsuccessful in computing a polynomial bound, and instead will offer a conjecture supported by analogy and evidence.

The analogy is with the hypercube. Recall that Fig. 1a has at least 2^k polygonizations, and these polygonization have a natural mapping to k-bit binary numbers. Moves between polygonizations are similar but not identical to bit flips. So a random walk through the polygonizations of this polygon is similar to a random walk on the hypercube graph.

We conducted an experiment to pursue this analogy, using a variant of the polygon shown in Fig. 15a, which breaks collinearities by distributing the vertices onto top, middle, and bottom circular arcs. We map each polygonization of this point set to a k-bit binary number, where the k^{th} bit indicates whether the shortest path from the k^{th} middle vertex

[5]The mixing time is roughly $\log N/(1-\mu)$, where μ is the mixing rate [Lov93, p. 358].

[6]See `http://theory.csail.mit.edu/~edemaine/polygonization/` for further references.

is to a top (1) or bottom (0) vertex.[7] (Note this map is many-to-one, as there are more than 2^k polygonizations.)

Although we do not have a precise method of generating forward and reverse moves with equal probability (to follow the equal-transitions model), we can simulate these moves by selecting a random stretch (recall that both forward and reverse moves commence with a stretch), followed by a sequence of twangs and stretches. To capture reverse twang cascades, we permit stretching from a vertex in multiple contact with a heuristically determined probability. Because forward moves always reduce the perimeter, we used long-term stability of the perimeter to indicate that the heuristic finds an equal balance of forward and reverse moves.

With these caveats noted, the results are displayed in Fig. 15b, which shows the number of the 256 bit patterns reached over 5000 stretches, overlayed with the number of bit patterns reached in a hypercube walk of 1000 bit flips. After 5000 stretches, 91% of the polygon patterns were visited (and 22 patterns were not reached). The hypercube walk reached 96% of the patterns after 1000 bit flips. Although differences in convergence are evident, it seems fair to say that the two walks (when scaled) are analogous.

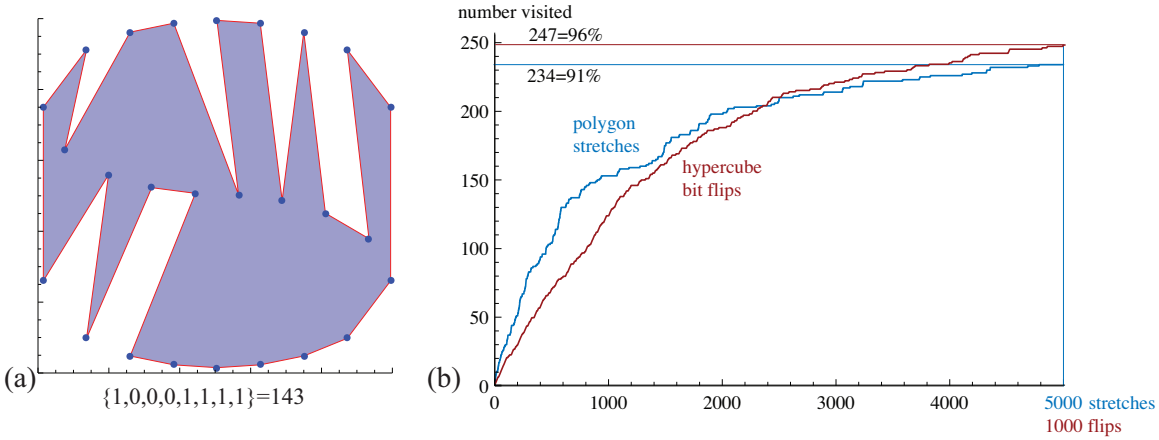

Figure 15: k=8, 2^k=256. (a) Polygonization→bits map. (b) Numbers visited vs. stretches, both for polygonizations and hypercube bit flips. The horizontal scale for polygonizations is 5× that for hypercube.

Each node of the hypercube graph on n bits has degree n, and we noted above that some nodes of G have degree $\Omega(n^2)$. We suspect that this difference will retard the mixing time in comparison to the hypercube's $O(n \log(n/\varepsilon))$, but we conjecture that random walks on G are still rapidly mixing, perhaps $O((n/\varepsilon)^2)$.

9. Open Problems

Our work leaves many interesting problems open. A central unresolved question is whether there is a combinatorial upper bound on the number of twangs T in a twang cascade. We have shown that T is $\Omega(n^2)$, but have no upper bound except that of Lemma 2.2.

[7]Path length is measured by the number of edges, with Euclidean length breaking ties.

We have established a few properties of the polygonization graph G (connectedness, nonbipartiteness, upper bounds on node degree and diameter), but its structure remains to be fully elucidated. For example, we have not established lower bounds on either node degree or diameter.

In Sec. 7 we established connectivity with forward moves and their reverse, and although both moves are composed of atomic stretches and twangs, the forward moves seem more naturally determined. This suggests the question of whether forward moves suffice to ensure connectivity.

Using stretches and twangs to generate random polygons (Sec. 8) raises many issues, the most prominent being settling our conjecture that the random walk through G is rapidly mixing.

Finally, we are extending our work to 3D polyhedralizations of a fixed 3D point set.

Acknowledgements. We are grateful to the referees for their insightful comments, suggestions, and corrections.

References

[AH96] Thomas Auer and Martin Held. Heuristics for the generation of random polygons. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 38–43, 1996.

[BB04] David Dai-Wai Bao and David Bao. *Sampler of Riemann-Finsler Geometry*. Cambridge University Press, 2004.

[BH08] Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Comput. Geom. Theory. Appl.*, 2008. To appear.

[CHUZ01] Jurek Czyzowicz, Ferran Hurtado, Jorge Urrutia, and Najib Zaguia. On polygons enclosing point sets. *Geombinatorics*, XI:21–28, 2001.

[DFOR08] Mirela Damian, Robin Flatland, Joseph O'Rourke, and Suneeta Ramaswami. Connecting polygonizations via stretches and twangs. In *Proc. 25th Sympos. Theoretical Aspects Comput. Sci. (STACS)*, pages 217–228. IBFI Schloss Dagstuhl, February 2008.

[GNT00] Alfredo García, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Comput. Geom. Theory Appl.*, 16:211–221, 2000.

[HHH02] Carmen Hernando, Ferran Hurtado, and Michael Houle. On local transformation of polygons with visibility properties. *Theoretical Computer Science*, 289:919–937, 2002.

[Kan94] Ravi Kannan. Markov chains and polynomial time algorithms. In *35th IEEE Sympos. Foundations Comput. Sci (FOCS)*, pages 656–671, 1994.

[Lov93] László Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdös is Eighty*, volume 2 of *Bolyai Society Math. Studies, 2*, pages 353–398. 1993.

[MRS99] Michael Molloy, Bruce Reed, and William Steiger. On the mixing rate of the triangulation walk. In *DIMACS Series in Disc. Math. and Theoret. Comput. Sci.*, volume 43, pages 179–190, 1999.

[Ran06] Dana Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science and Engg.*, 8(2):30–41, 2006.

[Sin93] Alistair Sinclair. *Algorithms for random generation and counting: a Markov chain approach*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993.

[vLS82] Jan van Leeuwen and Anneke A. Schoone. Untangling a travelling salesman tour in the plane. In J. R. Mühlbacher, editor, *Proc. 7th Internat. Workshop Graph-Theoret. Concepts Comput. Sci.*, pages 87–98, München, 1982. Hanser.

[ZSSM96] Chong Zhu, Gopalakrishnan Sundaram, Jack Snoeyink, and Joseph S. B. Mitchell. Generating random polygons with given vertices. *Comput. Geom. Theory Appl.*, 6:277–290, 1996.