Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 217-228
www.stacs-conf.org

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". These atomic moves 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 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 polygon: $P = P_\sigma = (p_{\sigma(0)}, \ldots, p_{\sigma(n-1)})$ is a simple (non-self-intersecting) polygon. 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 have as few as 1 polygonization, if S is in convex position,[1] and as many as $2^{\Theta(n)}$ polygonizations. For the latter, 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]:

1998 ACM Subject Classification: Nonnumerical Algorithms: F.2.2; Discrete Mathematics: G.2.
Key words and phrases: polygons, polygonization, random polygons, connected configuration space.
[1]S is in convex position if every point in S is on the hull of S.

> "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 determined by a sequence* σ *of point indices that includes every index in* $\{0, 1, \ldots, n-1\}$ *at least once, such that there is a perturbation of the points in multiple contact that renders* $\mathcal{P}_\sigma$ *a simple closed curve through the perturbed points in* σ *order.*

Thus polygonal wraps disallow proper crossings[2] 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. 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 loose bounds on the worst-case number of atomic operations as a function of n: an exponential upper bound and a quadratic lower bound. Tightening these bounds 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). 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 established 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 explore the possible application to random polygons briefly in Sec. 8. For the majority of this paper, we concentrate on defining the moves and establishing connectivity.

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 these preliminaries, we establish connectivity and compute the diameter in Secs. 3–7. We conclude with open problems in Sec. 9. Omitted proofs are in [DFOR07].

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

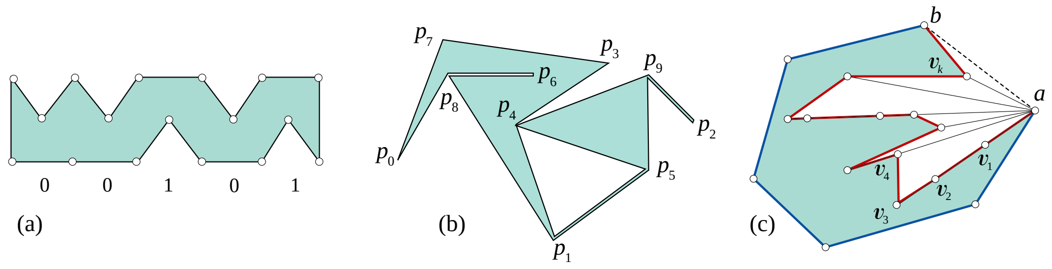

Figure 1: Examples. (a) A set of $n = 3k + 2$ points that admits 2^k polygonizations. (b) Polygonal wrap $\mathcal{P}_\sigma$ with $\sigma = (0, 8, 6, 8, 1, 5, 9, 2, 9, 4, 5, 1, 4, 3, 7)$ (c) A polygonization with one pocket with lid ab.

1.1. Pockets and Canonical Polygonization

Let P be a polygonization of S. A hull edge ab that is not on ∂P is called a *pocket lid*. The polygon external to P bounded by P and ab is a *pocket* of P. For a fixed hull edge ab, 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 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 $\textsc{Hop}(e, v)$, 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, $\textsc{Hop}(e, v)$ is valid iff 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).

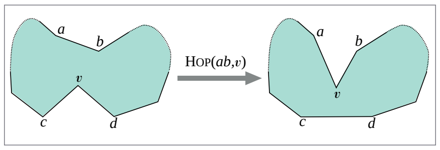

Figure 2: $\textsc{Hop}(ab, v)$ illustrated.

2.2. Stretches and Twangs

Unlike the $\textsc{Hop}(e, v)$ operation, which requires v to fully see the edge e into which it is hopping, the $\textsc{Stretch}(e, v)$ operation only requires that v see a point x in the interior[3] 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 $\textsc{Stretch}_0$ below), and (ii) remove the pseudovertices immediately using twangs.

$\textsc{Stretch}_0$. 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 clearly 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^- closer to the endpoint a of e. Then

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

$\text{Stretch}_0(e, v)$ 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).

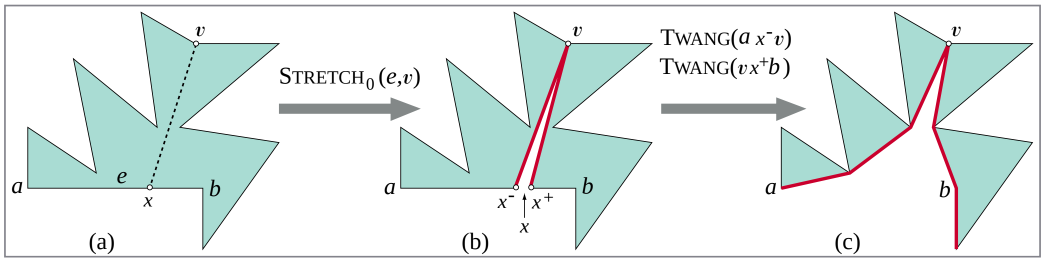

Figure 3: $\text{Stretch}(e, v)$ illustrated (a) v sees $x \in e$ (b) $\text{Stretch}_0(e, v)$ (c) $\text{Stretch}(e, v)$.

To complete the definition of $\text{Stretch}(e, v)$, 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 double contact. If b is a vertex in double contact, then $\triangle abc$ does not contain a* nested *double contact at b. By this we mean the following: Slightly perturb the vertices of P to separate each double-contact into two or more points, 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 $\triangle abc$ 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 by* Twang(b)*.*

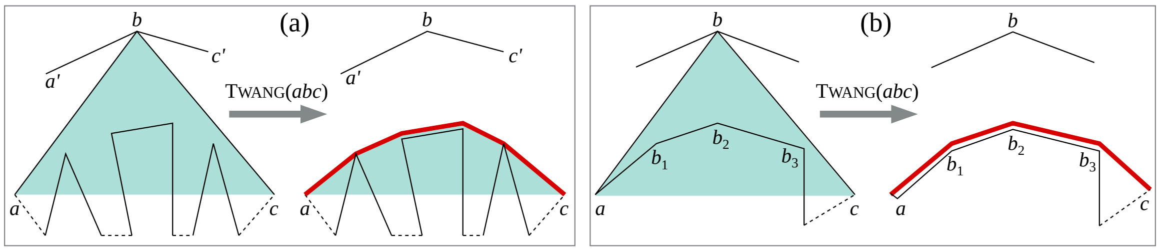

Figure 4: $\text{Twang}(abc)$ illustrated (a) $\text{Twang}(abc)$ replaces abc by sp(abc) (b) $\text{Twang}(abc)$ creates the hairpin vertex a and three doubled edges ab_1, b_1b_2 and b_2b_3.

Informally, $\text{Twang}(abc)$ "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 removing either a pseudovertex or a point of double contact. 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 "doubled 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 interior to the triangle, and the twang creates new points of double contact. Below, we will apply twangs repeatedly to remove all double 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 double contact and creates perhaps several others. A TWANGCASCADE applied on a polygonal wrap $\mathcal{P}$ removes all points of double contact from $\mathcal{P}$:

```
TWANGCASCADE($\mathcal{P}$)
Loop for as long as $\mathcal{P}$ has a point of double contact $b$:
    1. Find a vertex sequence $abc$ in $\mathcal{P}$ that satisfies the twang conditions (cf. Def. 2.1).
    2. TWANG($abc$).
```

Note that for any point b of double contact, there always exists a vertex sequence abc that satisfies the twang conditions and therefore the twang cascade loop never gets stuck. 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 sp(abc)) by at least a constant 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* $2d_{\min}(1 - \sin(\alpha_{\max}/2))$*, where* $d_{\min}$ *is the smallest pairwise point distance and* $\alpha_{\max}$ *is the maximum convex angle formed by any triple of non-collinear points.*

Supplementing this geometric bound, we establish in [DFOR07, App. 3] a combinatorial upper bound of $O(n^n)$ on the number of twangs in any twang cascade. An impediment to establishing a better bound is that a point can twang more than once in a cascade. Indeed we present an example in which $\Omega(n)$ points each twang $\Omega(n)$ times in one cascade, providing an $\Omega(n^2)$ lower bound.

2.3.1. *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 noncollinear 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. 5), 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. 5, where the initial Stretch(ab, v) inserts v between a and b in the polygonal wrap (Fig. 5b), but v ends up between c and b in the final polygonization (Fig. 5f). 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).

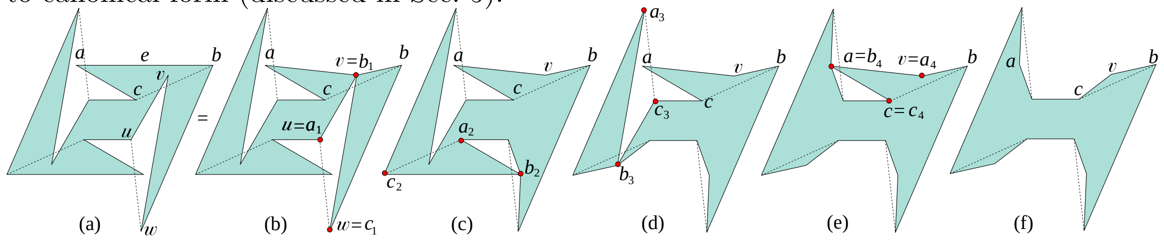

Figure 5: Forward move illustrated. (a) Initial polygon P (b) After Stretch(ab, v) (c) After Twang($a_1b_1c_1$) (d) After Twang($a_2b_2c_2$) (e) After Twang($a_3b_3c_3$) (f) After Twang($a_4b_4c_4$).

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 e that is the lid of a pocket A, the goal is to reduce A to e 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. 6 for an example run.

```
SINGLE POCKET REDUCTION$(P, e)$ Algorithm

Loop for as long as the pocket $A$ of $P$ with lid $e$ contains three or more vertices:
  1. Pick an edge-vertex pair $(e, v)$ such that
        $e$ is an edge of $P$ on $\partial B$ 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 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 sp$(abc) \subset \triangle abc$. □

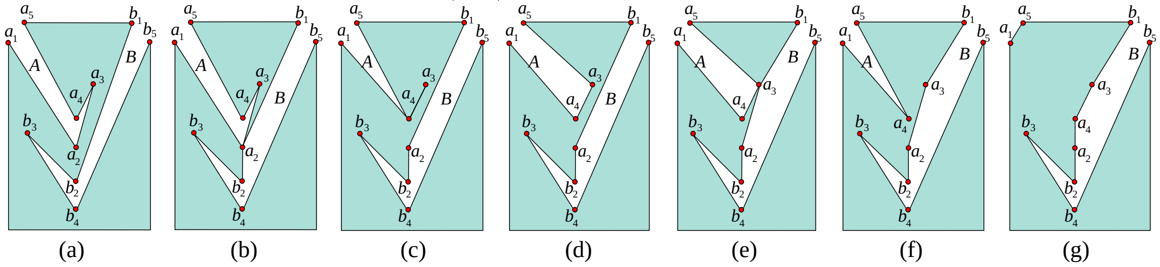

Figure 6: SINGLE POCKET REDUCTION(P, a_1a_5) illustrated: (a) Initial P; (b) After STRETCH(b_1b_2, a_2); (c) After TWANG$(a_1a_2a_3)$; (d) After TWANG$(a_3a_4a_5)$; (e) After STRETCH(a_2b_1, a_3); (f) After TWANG$(a_4a_3a_5)$; (g) After STRETCH(a_2a_3, a_4) + TWANG$(a_1a_4a_5)$.

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

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. 7 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_1a_5), 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. 6). Similarly, Single Pocket Reduction(P, b_1b_5) transfers to C all non-lid vertices of B, so C ends up with 9 non-lid vertices, and finally Single Pocket Reduction(P, c_1c_5) transfers all these vertices to T.

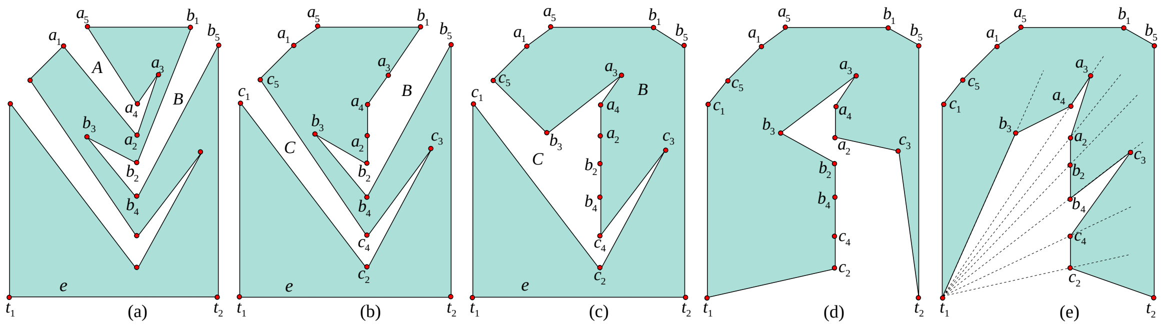

Figure 7: (a-e) Pocket Reduction(P, t_1t_2): (a) Initial P; (b) After Single Pocket Reduction(P, a_1a_5); (c) After Single Pocket Reduction(P, b_1b_5); (d) After Single Pocket Reduction(P, c_1c_5); (e) After Canonical Polygonization(P, t_1t_2).

This example shows that the $O(n^2)$ bound on the number of forward moves is tight: an n-vertex polygon with a structure similar to the one in Fig. 7a has $O(n)$ pockets. The number of forward moves performed by the Pocket Reduction algorithm is therefore $3 + 6 + 9 + \dots \frac{3n}{5} = \Theta(n^2)$, so we have the following lemma:

Lemma 4.1. *The* Pocket Reduction *algorithm employs* $\Theta(n^2)$ *forward moves.*

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) Algorithm

Let $e = ab$. Let $a = v_0, v_1, v_2, \dots, v_k, v_{k+1} = b$ be the canonical order of the vertices of pocket $P(e)$.
For each $i = 1, 2, \dots, k$
    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 [DFOR07, App. 1]:

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.*

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

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. 8a. 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.

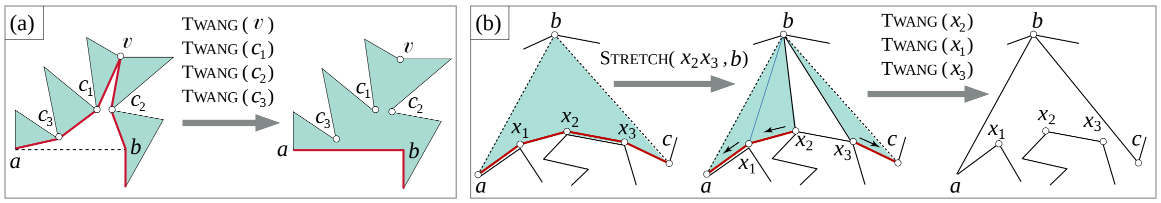

Figure 8: Reverse atomic moves: (a) Stretch(ab, v) is reversed by Twang(v), Twang(c_1), Twang(c_2), Twang(c_3). (b) Twang(b) is reversed by Stretch(x_2x_3, b), Twang(x_2), Twang(x_1) and Twang(x_3).

Reverse Twang. An "untwang" can be accomplished by one stretch followed by a series of twangs. Fig. 8b 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(e, 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 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. Reverse the order of the moves in $M_1 \oplus M_2$ ($\oplus$ represents concatenation).
5. For each stretch $s$ (twang $t$) in $M_1 \oplus M_2$ 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 $\Omega(n)$ and $O(n^2)$, and the diameter of the polygonization space is $O(n^2)$ moves.*

This diameter bound is tight for our specific algorithm 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 Polygons

We have implemented a version of random polygon generation. After creating an initial polygonization, we move from polygonization to polygonization via a sequence of forward moves, where additional stretches are permitted in the cascade to simulate reverse moves. Here we report on one experiment that investigates the speed with which the exponential space of polygonizations is explored. We use a variant of the example in Fig. 1a, which has at least 2^k polygonizations. The variant is shown in Fig. 9a, 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 is to a top (1) or bottom (0) vertex.[4] (Note this map is many-to-one, as there are more than 2^k polygonizations.) Starting from an arbitrary polygonization, we then repeatedly select a random stretch, and twang to quiescence. Figs. 9b,c display the range of the random walk in two formats: (b) shows the number of the 256 bit patterns reached over the 5000 stretches—91% of the patterns were visited by the end of the trial; (c) shows when each bit pattern was reached (dark), with time growing downwards. By the final stretch, 22 patterns (light) were yet to be visited. In this trial, the average length of a twang cascade was 1.2; more precisely, the 5000 stretches invoked 5960 twangs, for a total of 10,960 atomic moves.

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

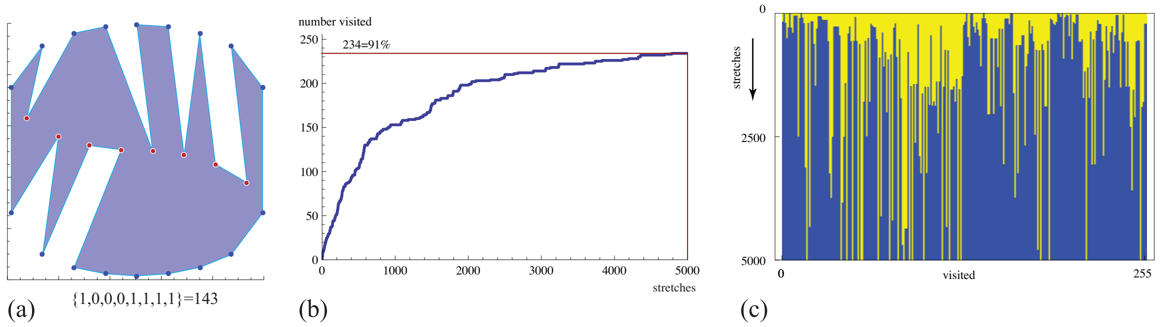

Figure 9: k=8, 2^k=256. (a) Polygonization→bits map. (b) Numbers visited vs. stretches. (c) Dark: numbers visited with increasing stretches; light: not yet visited.

9. Open Problems

Our work leaves many interesting problems open. One unresolved question is whether the number of twangs T in a twang cascade is exponential or if there is a polynomial bound, thereby resolving the computational complexity of the polygon transformation algorithm. We have shown that T is $\Omega(n^2)$ and $O(n^n)$, leaving a large gap to be closed. We would also like to establish a lower bound on the 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.

It remains to be seen if the polygonization moves explored in this paper will be effective tools for generating random polygons. One possibility is to start from a doubled random noncrossing spanning tree, which is a polygonal wrap. Finally, we are extending our work to 3D polyhedralizations of a fixed 3D point set.

References

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

[CHUZ01] J. Czyzowicz, F. Hurtado, J. Urrutia, and N. Zaguia. On polygons enclosing point sets. *Geombinatorics*, XI (1):21–28, 2001.

[DFOR07] M. Damian, R. Flatland, J. O'Rourke, and S. Ramaswami. Connecting polygonizations via stretches and twangs. `arxiv.org`, arXiv:0709.1942v1 [cs.CG], 2007.

[HHH02] C. Hernando, M. Houle, and F. Hurtado. On local transformation of polygons with visibility properties. *Theoretical Computer Science*, 289(2):919-937, 2002.

[vLS82] J. van Leeuwen and A. 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] C. Zhu, G. Sundaram, J. Snoeyink, and J. S. B. Mitchell. Generating random polygons with given vertices. *Comput. Geom. Theory Appl.*, 6:277–290, 1996.