Approximate Convex Decomposition of Polygons[*]

Jyh-Ming Lien   Nancy M. Amato

{neilien,amato}@cs.tamu.edu

Parasol Lab., Department of Computer Science

Texas A&M University

Abstract

We propose a strategy to decompose a polygon, containing zero or more holes, into *"approximately convex"* pieces. For many applications, the approximately convex components of this decomposition provide similar benefits as convex components, while the resulting decomposition is significantly smaller and can be computed more efficiently. Moreover, our *approximate convex decomposition* (ACD) provides a mechanism to focus on key structural features and ignore less significant artifacts such as wrinkles and surface texture. We propose a simple algorithm that computes an ACD of a polygon by iteratively removing (*resolving*) the most significant non-convex feature (*notch*). As a by product, it produces an elegant hierarchical representation that provides a series of 'increasingly convex' decompositions. A user specified tolerance determines the degree of concavity that will be allowed in the lowest level of the hierarchy. Our algorithm computes an ACD of a simple polygon with n vertices and r notches in $O(nr)$ time. In contrast, exact convex decomposition is NP-hard or, if the polygon has no holes, takes $O(nr^2)$ time. Models and movies can be found on our web-pages at: http://parasol.tamu.edu/groups/amatogroup/

Keywords: convex decomposition, hierarchical, polygon.

[*]This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0081510, ACR-

1 Introduction

Decomposition is a technique commonly used to break complex models into sub-models that are easier to handle. Convex decomposition, which partitions the model into convex components, is interesting because many algorithms perform more efficiently on convex objects than on non-convex objects. Convex decomposition has application in many areas including pattern recognition [17], Minkoski sum computation [1], motion planning [22], computer graphics [30], and origami folding [15].

One issue with convex decompositions, however, is that they can be costly to construct and can result in representations with an unmanageable number of components. For example, while a minimum set of convex components can be computed efficiently for simple polygons without holes [11, 12, 27], the problem is NP-hard for polygons with holes [33].

In this paper, we propose an alternative partitioning strategy that decomposes a given polygon, containing zero or more holes, into *"approximately convex"* pieces. Our motivation is that for many applications, the approximately convex components of this decomposition provide similar benefits as convex components, while the resulting decomposition is both significantly smaller and can be computed more efficiently. Features of this approach are that it

- applies to any simple polygon, with or without holes,
- provides a mechanism to focus on key features, and
- produces a hierarchical representation of convex decompositions of various levels of approximation.

Figure 1 shows an approximate convex decomposition with 128 components and a minimum convex decomposition with 340 components [27] of a Nazca line monkey.[†]

Our approach is based on the premise that for some models and applications, some of the non-convex (concave)

0113971, CCR-0113974, EIA-9810937, EIA-0079874, and by the Texas Higher Education Coordinating Board grant ATP-000512-0261-2001. A preliminary version of this work appeared in the Proceedings of the ACM Symposium on Computational Geometry 2004 [32].

[†]Nazca lines [8] are mysterious drawings found in southwest Peru. They have lengths ranging from several meters to kilometers and can only be recognized by aerial viewing. Two drawings, monkey and heron, are used as examples in this paper.

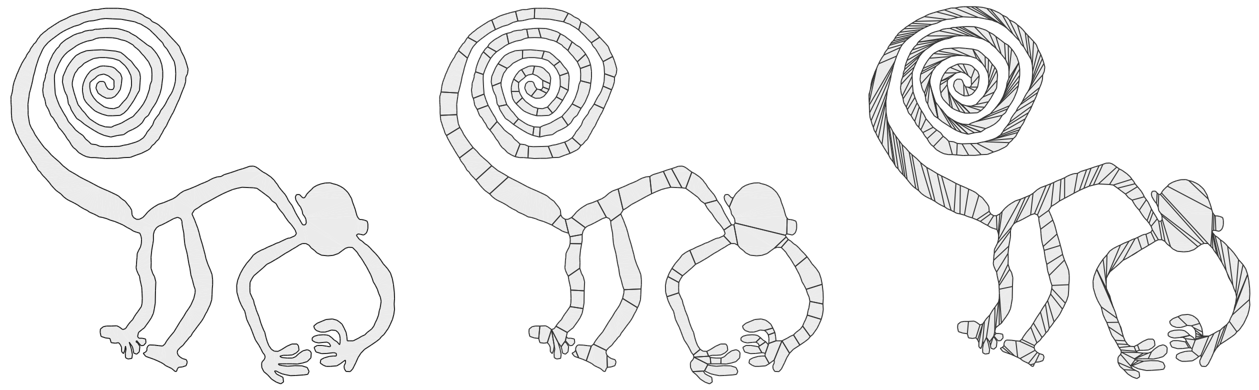
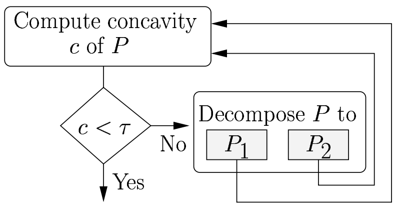


Figure 1: (a) The initial Nazca monkey has 1,204 vertices and 577 notches. The radius of the minimum bounding circle of this model is 81.7 units. Setting the concavity tolerance at 0.5 units, and not allowing Steiner points, (b) an approximate convex decomposition has 126 approximately convex components, and (c) a minimum convex decomposition has 340 convex components.



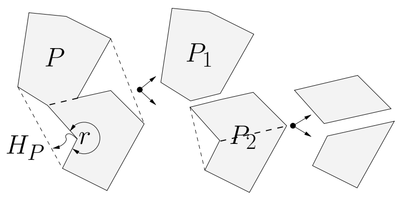


Figure 2: (a) Decomposition process. The tolerable concavity τ is user input. (b) A hierarchical representation of polygon P. Vertex r is a notch and concavity is measured as the distance to the convex hull H_P.

features can be considered *less significant*, and allowed to remain in the final decomposition, while others are more important, and must be removed (resolved). Accordingly, our strategy is to identify and resolve the non-convex features in order of importance. An overview of the decomposition process is shown in Figure 2(a). Due to the recursive application, the resulting decomposition has a natural hierarchy represented as a binary tree. An example is shown in Figure 2(b), where the original model P is the root of the tree, and its two children are the components P_1 and P_2 resulting from the first decomposition. If the process is halted before convex components are obtained, then the leaves of the tree are approximate convex components. Thus, the hierarchical representation computed by our approach provides multiple Levels of Detail (LOD). A single decomposition is constructed based on the highest accuracy needed, but coarser, "less convex" components can be retrieved from higher levels in the decomposition hierarchy when the computation does not require that accuracy.

For some applications, the ability to consider only important features may not only be more efficient, but may lead to improved results. In pattern recognition, for example, features are extracted from images and polygons to represent the shape of the objects. This process, e.g., skeleton extraction, is usually sensitive to small detail on the boundary, such as surface texture, which reduces the quality of the extracted features. By extracting a skeleton from the convex hulls of the components in an approximate decomposition, the sensitivity to small surface features can be removed, or at least decreased [30].

The success of our approach depends critically on the accuracy of the methods we use to prioritize the importance of the non-convex features. Intuitively, important features provide key structural information for *the application*. For instance, visually salient features are important for a visualization application, features that have significant impact on simulation results are important for scientific applications, and features representing anatomical structures are important for character animation tools. Although curvature has been one of the most popular tools used to extract visually salient features, it is highly unstable because it identifies features from local variations on the polygon's boundary. In contrast, the concavity measures we consider here identify features using global properties of the boundary. Figure 2(b) shows one possible way to measure the concavity of a polygon as the maximal distance from a vertex of P (r in this example) to the boundary of the convex hull of P. When the concavity (of a polygon P) obtained using a certain concavity measure is "small enough" to be ignored, P can be considered as convex or P can be represented by its convex hull. We say an approximate convex decomposition (or polygon) is *τ-convex* if all vertices in the decomposition (or polygon) have concavity less than τ.

The paper is organized as follows. We begin by defining the notation used in this paper in Section 2 and in Section 3 we review previous work on convex decomposition. Next, we present our approximate convex decomposition framework in Section 4. General ideas and details of our concavity measurements are presented in Section 5. In Section 6, we analyze the complexity of the method and provide implementation details and experiment results in Section 7.

2 Preliminaries

A polygon P is represented by a set of boundaries $\partial P = \{\partial P_0, \partial P_1, \ldots, \partial P_k\}$, where ∂P_0 is the external boundary and $\partial P_{i>0}$ are boundaries of *holes* of P. Each boundary ∂P_i consists of an ordered set of vertices V_i which defines a set of edges E_i. Figure 3(a) shows an example of a simple polygon with nested holes. A polygon is *simple* if no nonadjacent edges intersect. Thus, a simple polygon P with nested holes is the region enclosed in ∂P_0 minus the region enclosed in $\cup_{i>0}\partial P_i$. We note that nested polygons can be treated independently. For instance, in Figure 3(a), the region bounded by ∂P_0 and $\partial P_{1\leq i\leq 4}$ and the region bounded by ∂P_5 can be processed separately.

The *convex hull* of a polygon P, H_P, is the smallest convex set containing P. P is said to be *convex* if $P = H_P$. A polygon C is a component of P if $C \subset P$. A set of components $\{C_i\}$ is a *decomposition* of P if their union is P and all C_i are interior disjoint, i.e., $\{C_i\}$ must satisfy:

$$\mathrm{D}(P) = \{C_i \mid \cup_i C_i = P \text{ and } \forall_{i\neq j} C_i \cap C_j = \emptyset\}. \tag{1}$$

A *convex decomposition* of P is a decomposition of P that contains only convex components, i.e.,

$$\mathrm{CD}(P) = \{C_i \mid C_i \in \mathrm{D}(P) \text{ and } C_i = H_{C_i}\}. \tag{2}$$

Our concavity measures use the concepts of *notches*, *bridges* and *pockets*; see Figure 3(b). Vertices of P are *notches* if they have internal angles greater than 180°. Bridges are convex hull edges that connect two non-adjacent vertices of ∂P_0, i.e., $\mathrm{BRIDGES}(P) = \partial H_P \setminus \partial P$. Pockets are maximal chains of non-convex-hull edges of P, i.e., POCKETS$(P) = \partial P \setminus \partial H_P$. Observation 2.1 states the relationship between bridges, pockets, and notches.

Observation 2.1. *Given a simple polygon P. Notches can only be found in pockets. Each bridge has an associated pocket, the chain of ∂P_0 between the two bridge vertices. Hole boundaries are also pockets, but they have no associated bridge.*

3 Related Work

Many approaches have been proposed for decomposing polygons; see the survey by Keil [26]. The problem of convex decomposition of a polygon is normally subject to some optimization criteria to produce a minimum number of convex components or to minimize the sum of length of the boundaries of these components (called minimum ink [26]). Convex decomposition methods can be classified according to the following criteria:

- Input polygon: simple, holes allowed or disallowed.
- Decomposition method: Steiner points allowed or disallowed.
- Output decomposition properties: minimum number of components, shortest internal length, etc.

For polygons with holes, the problem is NP-hard for both the minimum components criterion [33] and the shortest internal length criterion [25, 34].

When applying the minimum component criterion for polygons without holes, the situation varies depending on whether Steiner points are allowed. When Steiner points are not allowed, Chazelle [9] presents an $O(n \log n)$ time algorithm that produces fewer than $4\frac{1}{3}$ times the optimal number of components, where n is the number of vertices. Later, Green [19] provided an $O(r^2 n^2)$ algorithm to generate the minimum number of convex components, where r is the number of notches. Keil [25] improved the running time to $O(r^2 n \log n)$, and more recently Keil and Snoeyink [27] improved the time bound to $O(n + r^2 \min(r^2, n))$. When Steiner points are allowed, Chazelle and Dobkin [12] propose an $O(n + r^3)$ time algorithm that uses a so-called X_k-pattern to remove k notches at once without creating any new notches. An X_k-pattern is composed of k segments with one common end point and k notches on the other end points.

When applying the shortest internal length criterion for polygons without holes, Greene [19] and Keil [24] proposed $O(r^2n^2)$ and $O(r^2n^2 \log n)$ time algorithms, respectively, that do not use Steiner points. When Steiner points are allowed, there are no known optimal solutions. An approximation algorithm by Levcopoulos and Lingas [29] produces a solution of length $O(p \log r)$, where p is the length of perimeter of the polygon, in time $O(n \log n)$.

Not all convex decomposition methods fall into the above classification. For example, instead of decomposing P into convex components whose union is P, Tor and Middleditch [44] "decompose" a simple polygon P into a set of convex components $\{C_i\}$ such that P can represented as $H_P - \cup_i C_i$, where "$-$" is the set difference operator, and instead of decomposing a polygon, Fevens et al. [18] partition a constrained 2D point set S into convex polygons whose vertices are points in S.

Recently, several methods have been proposed to partition at salient features of a polygon. Siddiqi and Kimia [38] use curvature and region information to identify *limbs* and *necks* of a polygon and use them to perform decomposition. Simmons and Séquin [39] proposed a decomposition using an *axial shape graph*, a weighted medial axis. Tănase and Veltkamp [45] decompose a polygon based on the events that occur during the construction of a straight-line skeleton. These events indicate the annihilation or creation of certain features. Dey et al. [16] partition a polygon into *stable manifolds* which are collections of Delaunay triangles of sampled points on the polygon boundary. Since these methods focus on visually important features, their applications are more limited than our approximately convex decomposition. Moreover, most of these methods require pre-processing (e.g., model simplification [23]) or post-processing (e.g., merging over-partitioned components [16]) due to boundary noise.

4 Approximate Decomposition

Research in Psychology has shown that humans recognize shapes by decomposing them into components [5, 35, 38, 40]. Therefore, one approach that may produce a natural visual decomposition is to partition at the most *visually noticeable features*, such as the most dented or bent area, or an area with branches. Our approach for approximate convex decomposition follows this strategy. Namely, we recursively remove (resolve) concave features in order of decreasing significance until all remaining components have concavity less than some desired bound. One of the key challenges of this strategy is to determine approximate measures of concavity. We consider this question in Section 5. In this section, we assume that such a measure exists.

More formally, our goal is to generate τ-*approximate* convex decompositions, where τ is a user tunable parameter denoting the non-concavity tolerance of the application. For a given polygon P, P is said to be τ-*approximate* convex if $\text{concave}(P) < \tau$, where $\text{concave}(\rho)$ denotes the concavity measurement of ρ. A τ-*approximate* convex decomposition of P, $\text{CD}_\tau(P)$, is defined as a decomposition that contains only τ-*approximate* convex components; i.e.,

$$\text{CD}_\tau(P) = \{C_i \mid C_i \in \text{D}(P) \text{ and } \text{concave}(C_i) \le \tau\}. \tag{3}$$

Note that a 0-approximate convex decomposition is simply an exact convex decomposition, i.e., $\text{CD}_{\tau=0}(P) = \text{CD}(P)$.

Algorithm 4.1 Approx_CD(P, τ)

Input. A polygon, P, and tolerance, τ.

Output. A decomposition of P, $\{C_i\}$, such that $\max\{\text{concave}(C_i)\} \le \tau$.

```
1: $c = \text{concave}(P)$
2: **if** $c.\text{value} < \tau$ **then**
3:     return $P$
4: **else**
5:     $\{C_i\}$=**Resolve**($P$, $c$.witness).
6:     **for** Each component $C \in \{C_i\}$ **do**
7:         Approx_CD($C$,$\tau$).
8:     **end for**
9: **end if**
```

Algorithm 4.1 describes a *divide-and-conquer* strategy to decompose P into a set of τ-*approximate* convex pieces. The algorithm first computes the concavity and a point $x \in \partial P$ witnessing it of the polygon P, i.e., x is one of the most concave features in P. If the concavity of P is within the specified tolerance τ, P is returned. Otherwise, if the concavity of P is above the maximum tolerable value, then the **Resolve**(P, x) sub-routine will remove the concave feature at x. A requirement of the **Resolve** subroutine is that if x is on a hole boundary (∂P_i, $i > 0$), then **Resolve** will merge the hole to the external boundary and if x is on the external boundary (∂P_0) then **Resolve** will split P into exactly two components. See Algorithm 4.2 and Figure 4(a) and (b). As described in Section 5, the way we measure concavity and implement **Resolve** ensures that this is the case. Our simple implementation of **Resolve** runs in $O(n)$ time. The process is applied recursively to all new components. The union of all components $\{C_i\}$ will be our final decomposition. The recursion terminates when the concavity of all components of P is less than τ. Note that the concavity of the features changes dynamically as the polygon is decomposed (see Figure 4(c)).

Algorithm 4.2 Resolve(P, r)

Input. A polygon, P, and a notch r of P.

Output. P with a diagonal added to r so that r is no longer a notch.

```
1: if $r \in \partial P_0$ then
2:     Add a diagonal $\overline{rx}$ according to Eq. 8, where $x$ is a vertex in $\partial P_0$.
3: else
4:     Add a diagonal $\overline{rx}$, where $x$ is the closest vertex to $r$ in $\partial P_0$.
5: end if
```

4.1 Selection of Non-Concavity Tolerance (τ)

The main task that still needs to be specified in Algorithm 4.1 is how to measure the concavity of a polygon. We use concavity measurement at a point as a primitive operation to decide whether a polygon P should be decomposed and to identify concave features of P. In principle, our approach should be compatible with reasonable measurement (the requirements for concavity measurement are discussed in Lemma 6.2 in Section 6), and indeed the selection of the measure for the non-concavity tolerance τ should depend on the application. For example, for some applications, such as shape recognition, it may be desirable for the decomposition to be scale invariant, i.e., the decompositions of two different sized polygons with the same shape should be identical. Measuring the distance from ∂P to ∂H_P is an example of measure that is not scale invariant because it would result in more components when decomposing a larger polygon. An example of a measure that could be scale invariant would be

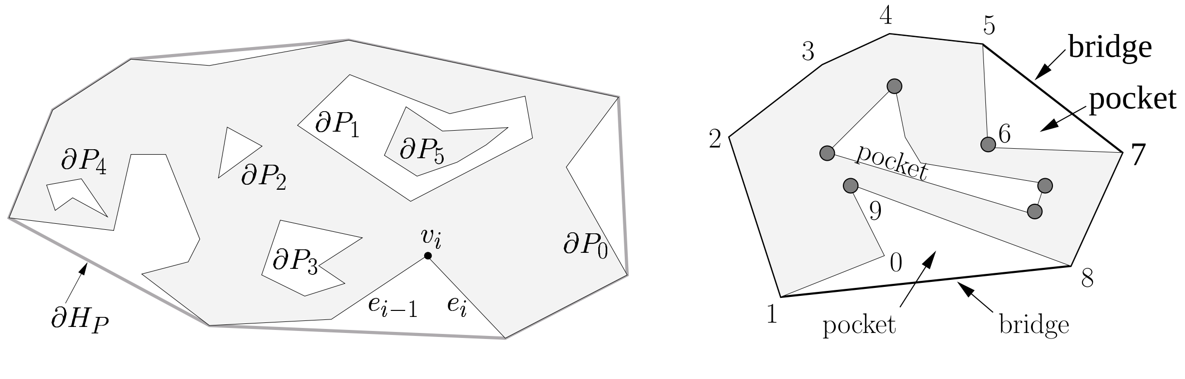


Figure 3: (a) A simple polygon with nested holes. (b) Vertices marked with dark circles are notches. Edges (5,7) and (8,1) are bridges with associated pockets $\{(5,6),(6,7)\}$ and $\{(8,9),(9,0),(0,1)\}$, respectively.

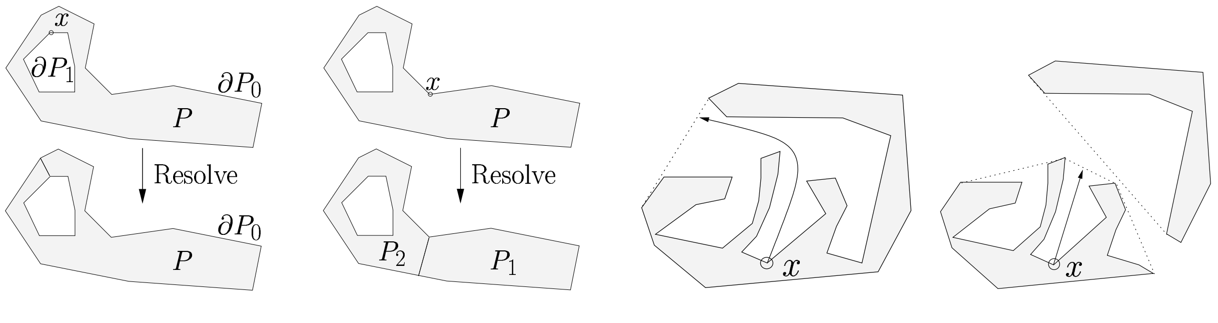


Figure 4: (a) If $x \in \partial P_{i>0}$, **Resolve** merges ∂P_i into P_0. (b) If $x \in \partial P_0$, **Resolve** splits P into P_1 and P_2. (c) The concavity of x changes after the polygon is decomposed.

a unitless measure of the similarity of the polygon to its convex hull. We present several methods for measuring concavity in Section 5.

5 Measuring Concavity

In contrast to measures like radius, surface area, and volume, concavity does not have a well accepted definition. For our work, however, we need a quantitative way to measure the concavity of a polygon. A few methods have been proposed [41, 7, 14, 6, 4] that attempt to measure the concavity of an image (pixel) based polygon as the distance from the boundary of P to the boundary of the pixel-based "convex hull" of P, called H'_P, using Distance Transform methods. Since P and H'_P are both represented by pixels, H'_P can only be nearly convex. *Convexity measurements* [43, 46] of polygons estimate the similarity of a polygon to its convex hull. For instance, the convexity of P can be measured as the ratio of the area of P to the area of the convex hull of P [46] or as the probability that a fixed length line segment whose endpoints are randomly positioned in the convex hull of P will lie entirely in P [46].

Another complication with trying to use a measure like convexity for our purposes is that since it is a global measure instead of a measure related to a feature of the polygon P, it is difficult to use convexity measurements to efficiently identify where and how to decompose a polygon so as to increase the convexity measurements. For example, Rosin [37] presents a shape partitioning approach that maximizes the convexity of the resulting components for a given number of cuts. His method takes $O(n^{2p})$ time to perform p cuts. This exponential complexity forbids any practical use of this algorithm in our case.

Although our approach is not restricted to a particular measure, all the measures we consider in this work define the concavity of a polygon as the maximum concavity of its boundary points, i.e., $\text{concave}(P) = \max_{x \in \partial P}\{\text{concave}(x)\}$. An important side effect of this decision is that now we can use points with with maximum concavity to identify important features where decomposition can occur. This would not be the case if we choose to sum concavities which would be similar to the convexity measurement in [43, 46]. An example illustrating this issue is shown in Figure 5.

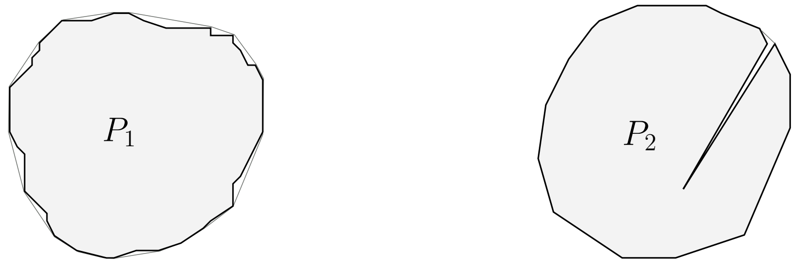
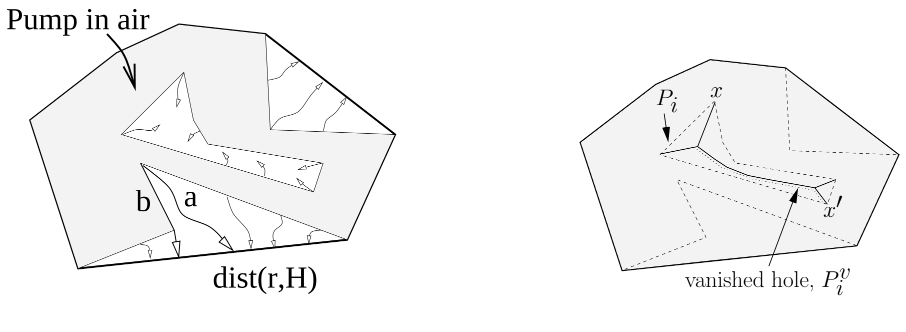


Figure 5: Although $\int_{\partial P_1} \mathrm{concave}(x)\,\mathrm{d}x = \int_{\partial P_2} \mathrm{concave}(x)\,\mathrm{d}x$, polygon P_1 is *visually* closer to being convex than polygon P_2.



(a) (b)

Figure 6: (a) The initial shape of a non-convex balloon (shaded). The bold line is the convex hull of the balloon. When we inflate the balloon, points not on the convex hull will be pushed toward the convex hull. Path a denotes the trajectory with air pumping and path b is an approximation of a. (b) The hole vanishes to its medial axis and vertices on the hole boundary will never touch the convex hull.

5.1 Measuring Concavity for External Boundary (∂P_0) Points

An intuitive way to define concave(x) for a point $x \in \partial P$ is to consider the trajectory of x when x is retracted from its original position to ∂H_P. More formally, let retract$(x, H_P, t) : \partial P \to H_P$ denote the function defining the trajectory of a point $x \in \partial P$ when x is retracted from its original position to ∂H_P. When $t = 0$, retract$(x, H_P, 0)$ is x itself. When $t = 1$, retract$(x, H_P, 1)$ is the final position of x on ∂H_P. Assuming that this retraction exists for x, concave(x) = dist(x, H_P) is the length of the function retract(x, H_P, t) from $t = 0$ to 1. In Section 6, we will provide a formal definition of the properties that we require for the retraction function. An intuition of this retraction function is illustrated in Figure 6(a). Think of P as a balloon which is placed in a mold with the shape of H_P. Although the initial shape of this balloon is not convex, the balloon will become so if we keep pumping air into it. Then the trajectory of a point on P to H_P can be defined as the path traveled by a point from its position on the initial shape to the final shape of the balloon. Although the intuition is simple, a retraction path such as path a in Figure 6(a) is not easy to define or compute.

Below, we describe three methods for measuring an approximation of this retraction distance that can be used in Algorithm 4.1. Recall that each pocket ρ in ∂P_0 is associated with exactly one bridge β. In Section 5.1.1, this retraction distance is measured by computing the straight-line distance from x to the bridge. Although this distance is fairly easy to compute, as we will see in Section 5.1.1, using it we cannot guarantee that the concavity of a point will decrease monotonically. A method that does not have this drawback is shown in Section 5.1.2, where we use the shortest path from x to the bridge in a visibility tree computed in the pocket. Unfortunately, this distance is more expensive to compute. Hybrid approaches that seek the advantages of both methods are proposed in Section 5.1.3.

5.1.1 Straight Line Concavity (SL-Concavity)

In this section, we approximate the concavity of a point x on ∂P_0 by computing the straight-line distance from x to its associated bridge β, if any. Note that this straight line may intersect P. Table 1 shows the decomposition of a Nazca monkey using SL-concavity.

Although computing the straight line distance is simple and efficient, this approach has the drawback of potentially leaving certain types of concave features in the final decomposition. As shown in Figure 7, the concavity of s does not decrease monotonically during the decomposition. This results in the possibility of leaving important features, such as s, hidden in the resulting components. This deficiency is also shown in the first image of Table 1 ($\tau = 40$) when the spiral tail of the monkey is not well decomposed. These artifacts result because the straight line distance does not reflect our intuitive definition of concavity.

5.1.2 Shortest Path Concavity (SP-Concavity)

In our second method, we find a shortest path from each vertex x in a pocket ρ to the bridge line segment $\beta = (\beta^-, \beta^+)$ such that the path lies entirely in the area enclosed by β and ρ, which we refer to as the *pocket polygon* and denote by P_ρ. Note that P_ρ must be a simple polygon. See Figure 8(a). In the following, we use $\pi(x, y)$ to denote the shortest path in P_ρ from an object x to an object y, where x and y can be edges or vertices. Two objects x and y are said to be *weakly visible* [3] to each other if one can draw at least one straight line from a point in x to a point in y without intersecting the boundary of P_ρ. A point x is said to be *perpendicularly visible* from a line segment β if x is weakly visible from β and one of the visible lines between x and β is perpendicular to β. For instance, points a and c in Figure 8(b) are perpendicularly visible from the bridge β and b and d are not. We denote by V_β^+ the ordered set of vertices that are perpendicularly visible from β, where vertices in V_β^+ have the same order as those in ∂P_0.

We compute the shortest distance to β for each vertex x in ρ according to the process sketched in Algorithm 5.1. First, we split P_ρ into three regions, $P_{\rho\beta^-}$, $P_{\rho\beta}$, and $P_{\rho\beta^+}$ as shown in Figure 8(b). The boundaries between $P_{\rho\beta^-}$ and $P_{\rho\beta}$ and $P_{\rho\beta}$ and $P_{\rho\beta^+}$, i.e., $\overline{a\beta^-}$ and $\overline{c\beta^+}$, are perpendicular to β. As shown in Lemma 5.2, the shortest paths for vertices x in $P_{\rho\beta^-}$ or $P_{\rho\beta^+}$ to β are the shortest paths to β^- or β^+, respectively. These paths can be found by constructing a *visibility tree* [20] rooted at β^- (β^+) to all vertices in $P_{\rho\beta^-}$ ($P_{\rho\beta^+}$).

The shortest path for a vertex $x \in B$ to β is composed of two parts: the shortest path $\pi(x, y)$, from x to some point y perpendicular visible to β, i.e., $y \in V_\beta^+$, and the $\pi(y, \beta)$ which is the straight line segment connecting y to β. Let $V_\beta^- = \{v \in \partial B\} \setminus V_\beta^+$. Figure 8(c) illustrates an example of V_β^+ and V_β^-. For each $v \in V_\beta^+$, there exists a subset of vertices in V_β^- that are closer to v than to any other vertices in V_β^+. These vertices must have shortest paths passing through v. For instance, in Figure 8(c), v_8 and v_7 must pass through v_6. Moreover, these vertices can be found by traversing the vertices of ∂B in order. For example, vertices between v_6 and v_{10} must have shortest paths passing through either v_6 or v_{10}.

Algorithm 5.1 SP_Concavity(β,ρ)

```
1: Split $P_\rho$ into polygons $P_{\rho\beta^-}$, $P_{\rho\beta}$, and $P_{\rho\beta^+}$ as shown in Figure 8(b).
2: Construct two visibility trees, $T^-$ and $T^+$, rooted in $\beta^-$ and $\beta^+$, respectively, to all vertices in $\rho$.
3: Compute $\pi(v,\beta)$, $\forall v \in P_{\rho\beta^-}$ (resp., $P_{\rho\beta^+}$) from $T^-$ (resp., $T^+$).
4: Compute an ordered set, $V_\beta^+$, in $P_{\rho\beta}$ from $T^-$ and $T^+$.
5: for each pair $(v_i, v_j) \in V^+(\beta)$ do
6:    for $i < k < j$ do
7:       $\pi(v_k,\beta) = \min\Big(\pi(v_k,v_i) + \pi(v_i,\beta), \pi(v_k,v_j) + \pi(v_j,\beta)\Big)$.
8:    end for
9: end for
10: Return $\{x, c\}$, where $x \in \rho$ is the farthest vertex from $\beta$ with distance $c$.
```

We compute V_β^+ by first finding vertices in $P_{\rho\beta}$ that are weakly visible from β and then filtering out vertices that are not perpendicularly visible fromβ. If a vertex $v \in B$ is weakly visible from β, both $\pi(v,\beta^-)$ and $\pi(v,\beta^+)$ must be *outward convex*. Following Guibas et al. [20], we say that $\pi(v,\beta^-)$ is outward convex if the convex angles formed by successive segments of this path keep increasing. Lemma 5.1 [20] states the property of two weakly visible edges. Our problem is a degenerate case of Lemma 5.1 as one of the edges collapses into a vertex, v. Therefore, finding weakly visible vertices of β can be done by constructing two visibility trees rooted at β^- and β^+.

Lemma 5.1. [20] *If edge $\overline{ab}$ is weakly visible from edge $\overline{cd}$, the two paths $\pi(a,c)$ and $\pi(b,d)$ are outward convex.*

The following lemma shows that Algorithm 5.1 finds the shortest paths from all vertices in the pocket ρ to its associated bridge line segment β.

Lemma 5.2. *Algorithm 5.1 finds the shortest path from every vertex v in pocket ρ to the bridge β.*

Proof. First we show that, for vertices v in region $P_{\rho\beta^-}$, $\pi(v,\beta)$ must pass through β^- to reach β. If the shortest path $\pi(v, \beta)$ from some $v \in A$ does not pass through β^- then it must intersect $\overline{\beta^- a}$ at some point which we denote $\hat{a}$. Vertex v_3 in Figure 8(c) is an example of such a vertex. However, the shortest path from $\hat{a}$ to β is the line segment from $\hat{a}$ to β^-. This contradicts the assumption that $\pi(v, \beta)$ does not pass through β^-. Therefore, all points in $P_{\rho\beta^-}$ must have shortest paths passing through β^-. Also, it has been proved that the visibility tree contains the shortest paths [28] from one vertex to all others in a simple polygon. Therefore, Line 3 in Algorithm 5.1 must find shortest paths to β for all vertices in $P_{\rho\beta^-}$. Similarly, it can be shown that $\pi(v, \beta)$ for all vertices in region $P_{\rho\beta^+}$ must pass through β^+.

For vertices v in region $P_{\rho\beta}$, we show that $\pi(v, \beta)$ must pass through V_β^+ to reach β. If $v \in V_\beta^+$, then the condition is trivially satisfied. Hence we need only consider $v \in V_\beta^-$. Vertices $v_8 \in V_\beta^-$ and $v_6 \in V_\beta^+$ in Figure 8(c) are examples of such vertices. If the shortest path $\pi(v, \beta)$ for some $v \in V_\beta^-$ does not pass through V_β^+ then it must intersect the segment perpendicular to β passing by some vertex in V_β^+. Let $v' \in V_\beta^+$ be the first such vertex and denote the point where $\pi(v, \beta)$ intersects $\perp v'\beta$ as $\hat{b}$. Since the shortest path from $\hat{b}$ to β is a straight line to β and it passes through $v' \in V_\beta^+$, we have a contradiction to the assumption that $\pi(v, \beta)$ does not pass through some $v \in V_\beta^+$. Therefore, Algorithm 5.1 must find the shortest path to β for all vertices in $P_{\rho\beta}$. $\square$

The concavity of a vertex v is the length of the shortest path from v to its associated bridge β. To compute the SP-concavity of ∂P_0, we find all bridge/pocket pairs and apply Algorithm 5.1 to each pair. Examples of retraction trajectories using SP-concavity are shown in Figure 9.

Next, we show that concave(P) decreases monotonically in Algorithm 4.1 if we use the shortest path distance to measure concavity. The guarantee of monotonically decreasing concavity eliminates the problem of leaving important concave features untreated as may happen using SL-concavity (see Table 1).

Lemma 5.3. *The concavity of ∂P_0 decreases monotonically during the decomposition in Algorithm 4.1 if we use SP-concavity.*

Proof. We show that the concavity of a point x in a pocket ρ of ∂P_0 either decreases or remains the same after another point $x' \in \rho$ is resolved. Let β be ρ's bridge with β^- and β^+ as end points. After x' is resolved, ρ breaks into two polygonal chains, from β^- to x' and from x' to β^+. New pockets and bridges will be constructed for both polygonal chains. Since the shortest path from x to the previous bridge β must intersect the bridge for x's new pocket, the new concavity of x will decrease or remain the same. $\square$

Finally, we show that Algorithm 5.1 takes $O(n)$ time to compute SP-concavity for all vertices on ∂P_0.

Lemma 5.4. *Measuring the concavity of the vertices on the external boundary ∂P_0 using shortest paths takes $O(n)$ time, where n is the size of ∂P_0.*

Proof. For each bridge/pocket, we show that the SP-concavity of all pocket vertices can be computed in linear time, which implies that we can measure the SP-concavity of P in linear time. First, it takes $O(n)$ time to split P into $P_{\rho\beta^-}$, $P_{\rho\beta}$, and $P_{\rho\beta^+}$ by computing the intersection between the pocket ρ and two rays perpendicular to β initiating from β^- and β^+. Then, using a linear time triangulation algorithm [10, 2], we can build a visibility tree in $O(n)$ time. Finding $V^+(\beta)$ takes $O(n)$ time as shown in [20]. The loop in Lines 5 to 8 of Algorithm 5.1 takes $\sum |j-i| \leq n = O(n)$ time since all (i,j) intervals do not overlap. Thus, Algorithm 5.1 takes $O(n)$ time and therefore we can measure the SP-concavity of P in $O(n)$ time. $\square$

5.1.3 Hybrid Concavity (H-Concavity)

We have considered two methods for measuring concavity: SL-concavity, which can be computed efficiently, and SP-concavity, which can guarantee that concavity decreases monotonically during the decomposition process. In this section, we describe a hybrid approach, called H-concavity, that has the advantages of both methods — SL-concavity is used as the default, but SP-concavity is used when SL-concavity would result in non-monotonically decreasing concavity of P.

SL-concavity can fail to report a significant feature x when the straight-line path from x to its bridge β intersects ∂P_0. In this case, x's concavity is under measured. Whether a pocket can contain such points can be detected by comparing the directions of the outward surface normals for the vertices v_i in the pocket and the outward normal direction $\vec{n}_\beta$ of the bridge β. The normal direction of a vertex v_i is the outward normal direction of the incident edge e_i; see Figure 10. The decision to use SL-concavity or SP-concavity is based on the following observation.

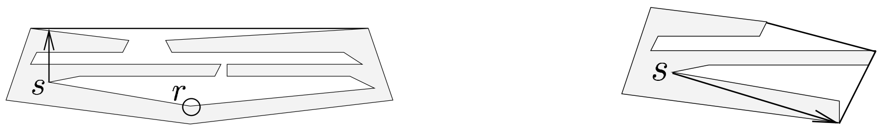


Figure 7: Let r be the notch with maximum concavity. After resolving r, the concavity of s increases. If concave(r) is less than τ, s will never be resolved even if concave(s) is actually larger then τ.

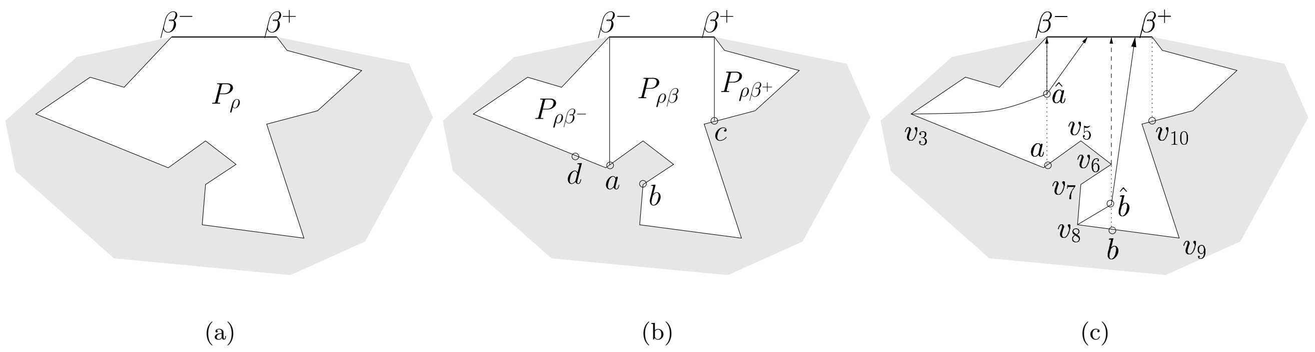


Figure 8: (a) P_ρ is a simple polygon enclosed by a bridge β and a pocket ρ. (b) Split P_ρ into $P_{\rho\beta^-}$, $P_{\rho\beta}$, and $P_{\rho\beta^+}$. (c) $V^-(\beta) = \{v_7, v_8, v_9\}$ and $V^+(\beta) = \{v_5, v_6, v_{10}\}$.

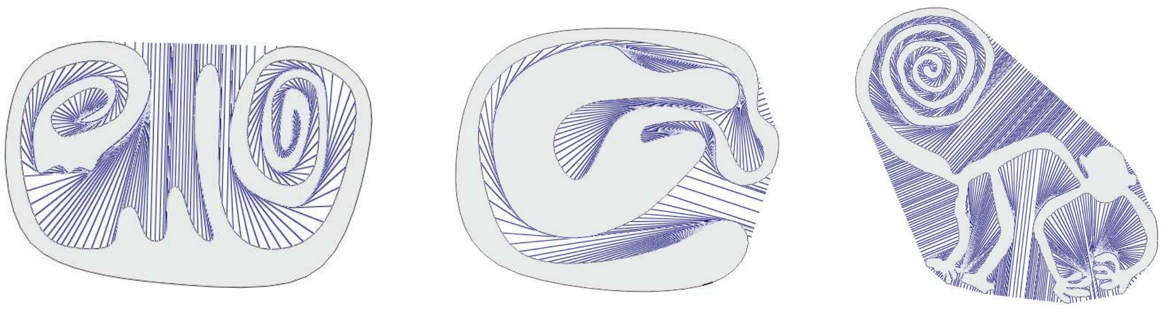


Figure 9: Shortest paths to the boundary of the convex hull.

Observation 5.5. *Let β and ρ be a bridge and pocket of ∂P_0, respectively. If* concave(∂P) *does not decrease monotonically using the SL-concavity measure, there must be a vertex $r \in \rho$ such that the normal vector of r, $\vec{n}_r$, and the normal vector of β, $\vec{n}_\beta$, point in opposite directions, i.e., $\vec{n}_r \cdot \vec{n}_\beta < 0$.*

This observation leads to Algorithm 5.2. We first use Observation 5.5 to check if SL-concavity can be used. If so, the concavity of P and its witness is computed using SL-concavity. Otherwise, SP-concavity is used. This approach improves the computation time and guarantees that the decomposition process has monotonically decreasing concavity.

Another option is to use SL-concavity more aggressively to compute the decomposition even more efficiently. This approach is described in Algorithm 5.3. First, we use SL-concavity to measure the concavity of a given bridge-pocket pair. If the maximum concavity is larger than the tolerance value τ, we split P. Otherwise, using Observation 5.5, we check if there is a possibility that some feature with untolerable concavity is hidden inside the pocket. If we find a potential violation, then SP-concavity is used. This approach is more efficient because it only uses SP-concavity if SL-concavity does not identify any untolerable concave features. We refer to the concavities computed using Algorithm 5.2 and Algorithm 5.3 as H1-concavity and H2-concavity, respectively.

Unlike H1-concavity, decomposition using H2-concavity may not have monotonically decreasing concavity. Thus, the order in which the concave features are found for H1- and H2-concavity can be different. Table 1 shows the decomposition process using H1-concavity and H2-concavity, respectively. The decomposition using H1-concavity is identical to that using SP-concavity. The decomposition using H2-concavity is more similar to the decompositions that would result from using SP-concavity with a larger τ or from using SL-concavity with smaller τ. We also observe that the relative computation costs of the different measures are, from slowest to fastest: SP-concavity, H1-concavity, H2-concavity, and finally SL-concavity. Experiments comparing decompositions using these concavity measures are presented in Section 7.

5.2 Measuring the Concavity for Hole Boundary ($\partial P_{i>0}$) Points

Note that in the balloon expansion analogy, points on hole boundaries will never touch the boundary ∂H_P of the convex hull H_P. The concavity of points in holes is therefore defined to be infinity and so we need some

Algorithm 5.2 H1-Concavity(β,ρ)

```
1: if No potential hazard detected, i.e., $\nexists r \in \rho$ such that $\vec{n}_r \cdot \vec{n}_\beta < 0$ then
2:     Return SL-concavity and its witness. (Section 5.1.1)
3: else
4:     Return SP-concavity and its witness. (Section 5.1.2)
5: end if
```

Algorithm 5.3 H2-Concavity(β,ρ)

```
1: SL-concavity and its witness $\{x, c\}$. (Section 5.1.1)
2: if $c > \tau$ then
3:     Return $\{x, c\}$.
4: end if
5: if No potential hazard detected, i.e., $\nexists r \in \rho$ such that $\vec{n}_r \cdot \vec{n}_\beta < 0$ then
6:     Return $\{x, c\}$.
7: end if
8: Return SP-concavity and its witness. (Section 5.1.2)
```

other measure for them. We will estimate the concavity of a hole P_i locally, i.e., without considering the external boundary ∂P_0 or the convex hull ∂H_p. Using the balloon expansion analogy again, we observe the following.

Observation 5.6. *P_i will "vanish" into a set of connected curved segments forming the medial axis of the hole as it contracts when ∂P_0 transforms to H_P. These curved segments will be the union of the trajectories of all points on ∂P_i to H_P once ∂P_i is merged with ∂P_0. Figure 6(b) shows an example of a vanished hole.*

5.2.1 Concavity for Holes

Recall that, from Observation 2.1, ∂P_i can also be viewed as a pocket without a bridge. The bridge will become known when a point $x \in \partial P_i$ is resolved, i.e., when a diagonal between x and ∂P_0 is added which will make ∂P_i become a pocket of ∂P_0. If x is resolved, the concavity of a point y in ∂P_i is concave(x) + dist(x, y). We define the *concavity witness* of x, $cw(x)$, to be a point on ∂P_i such that $\text{dist}(x, cw(x)) > \text{dist}(x, y)$, $\forall y \neq cw(x) \in \partial P_i$. That is, if we resolve x, then $cw(x)$ will be the point with maximum concavity in the pocket ∂P_i. Note that x and $cw(x)$ are associative, i.e., $cw(cw(x)) = x$, so that if we resolve $cw(x)$, x will be the point with maximum concavity in the pocket ∂P_i. See Figure 11. Intuitively, the maximum $\text{dist}(p, cw(p))$, where $p \in \partial P_i$ represents the "diameter" of P_i. The *antipodal pair* p and $cw(p)$ of the hole P_i represent important features because p (or $cw(p)$) will have the maximum concavity on ∂P_i when $cw(p)$ (or p) is resolved. Our task is to find p and $cw(p)$.

A naive approach to find the antipodal pair p and $cw(p)$ of P_i is to exhaustively resolve all vertices in ∂P_i. Unfortunately, this approach requires $O(n^2)$ time, where n is the number of vertices of P. Even if we attempt to measure the concavity of P_i locally without considering ∂P_0 and H_P, computing distances between all pairs of points in ∂P_i has time complexity $O(n_i^2)$, where n_i is the number of vertices of P_i.

5.2.2 Approximate Antipodal Pair, p and $cw(p)$

Fortunately, there are some possibilities to approximate p and $cw(p)$ more efficiently. As previously mentioned, in our balloon expansion analogy, a hole will contract to the medial axis which is a good candidate to find p and $cw(p)$ because it connects all pairs of points in the hole P_i. Once ∂P_i is merged to ∂P_0, concavity can be computed easily from the trajectories in the medial axis. Since P_i is a simple polygon, the medial axis of P_i forms a tree and can be computed in linear time [13]. We can approximate p and $cw(p)$ as the two points at maximum distance in the tree, which can be found in linear time.

Another way to approximate p and $cw(p)$ is to use the Principal Axis (PA) of P_i. The PA for a given set of points S is a line ℓ such that total distance from the points in S to ℓ is minimized over all possible lines $\kappa \neq \ell$, i.e.,

$$\sum_{x \in S} \text{dist}(x, \ell) < \sum_{x \in S} \text{dist}(x, \kappa), \ \forall \kappa \neq \ell. \tag{4}$$

In our case, S is the vertices of P_i. The PA can be computed as the Eigenvector with the largest Eigenvalue from the covariance matrix of the points in S. Once the PA is computed, we can find two vertices of P_i in two extreme directions on PA, and select one as p and the other as $cw(p)$. This approximation also takes $O(n)$ time.

Concavity measured using the PA resembles SL-concavity because in both cases concavity is measured as straight line distance and can be used when SL-concavity is desired. However, using the PA to measure SP-concavity can result in an arbitrary large error; see Figure 12(a). Thus, when SP-concavity is desired, concavity should be measured using the medial axis.

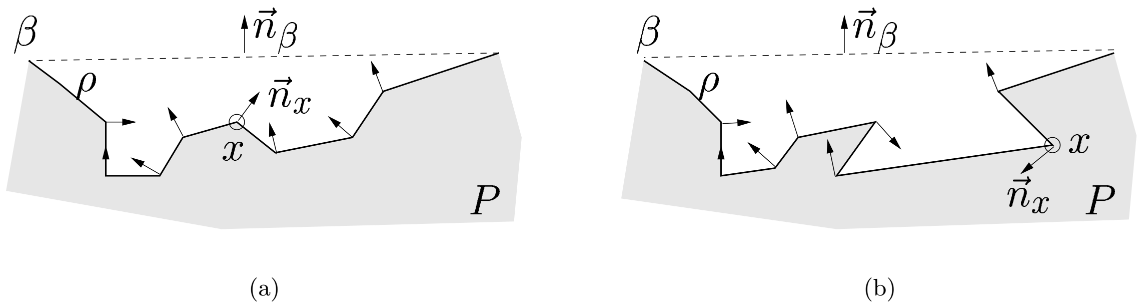


Figure 10: SL-concavity can handle the pocket in (a) correctly because none of the normal directions of the vertices in the pocket are opposite to the normal direction of the bridge. However, the pocket in (b) may result in non-monotonically decreasing concavity.

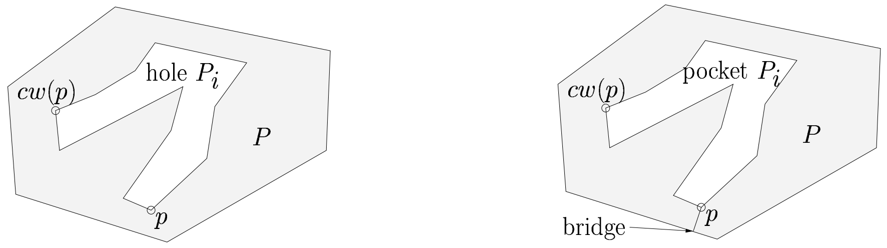


Figure 11: An example of a hole P_i and its antipodal pair. The maximum distance between p and $cw(p)$ represents the diameter of P_i. After resolving p, P_i becomes a pocket and $cw(p)$ is the most concave point in the pocket.

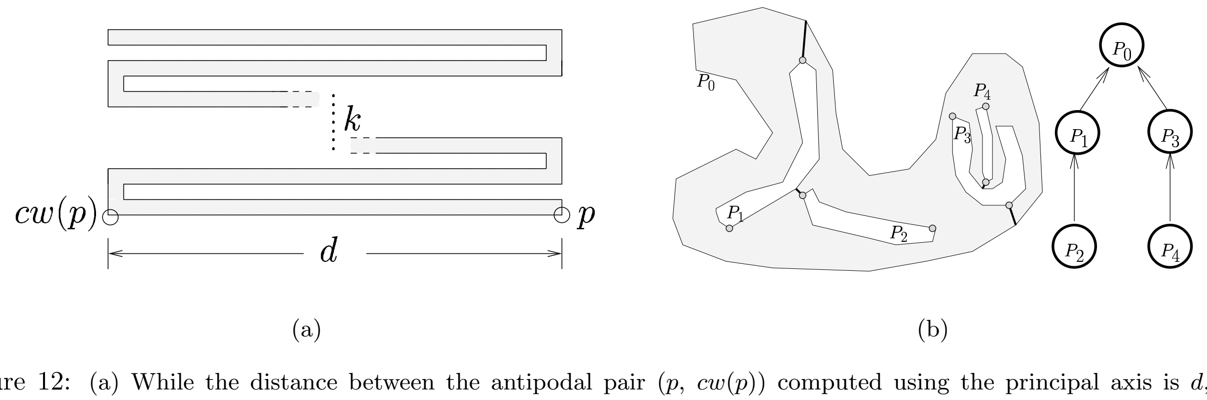


Figure 12: (a) While the distance between the antipodal pair (p, $cw(p)$) computed using the principal axis is d, the diameter of the hole with k turns is larger than $k \times d$. Note that k can be arbitrary large. (b) An example of hole resolution. Holes and the external boundary form a dependency graph which determines the order of resolution. In this case holes P_1 and P_3 will be resolved before P_2 and P_4. Dots on the hole boundaries are the antipodal pairs of the holes.

5.2.3 Measuring and Resolving Hole Concavity

For a polygon with k holes, we compute the antipodal pair, p_i and $cw(p_i)$, for each hole P_i, $1 \le i \le k$. A hole P_i is resolved when a diagonal is added between p_i and ∂P_0. Let x be a vertex of P closest to p_i (or $cw(p_i)$) but not in P_i. Without loss of generality, assume p_i is closer to x than $cw(p_i)$. We define the concavity of a hole P_i to be:

$$\text{concave}(P_i) = \text{concave}(x) + \text{dist}(x, p_i) + \text{dist}(p_i, cw(p_i)) + \delta \tag{5}$$

Since all vertices in a hole have infinite concavity, the term δ is defined as $concave(P_0)$ in Eq. 5 to ensure that hole concavity is larger than the concavity of P_0, and $\text{concave}(x) + \text{dist}(x, p_i)$ measures how "deep" the hole is from ∂P_0. If $x \in \partial P_0$, $\text{concave}(x)$ is already known. Otherwise, x is a vertex of a hole boundary $P_{j \neq i}$ and $\text{concave}(x) = \text{concave}(P_j)$. Note that this concavity definition implies the order of resolution of holes. An example is shown in Figure 12(b). Because x is the closest vertex to p_i, the line segment $\overline{p_i x}$ will not intersect anything.

6 Analysis

In Algorithm 4.1, we first find the most concave feature, i.e., the point $x \in \partial P$ with maximum concavity, and remove that feature x from P. In this section, we show that x must be a notch (Lemma 6.2) and that if the tolerable concavity is zero then the result will be an exact convex decomposition, i.e., all notches must be removed (Lemma 6.3). First, observe that if x is a notch, then the concavity of x must be larger than zero.

Lemma 6.1. *If a point $r \in \partial P$ is a notch,* $\text{concave}(r)$ *is not zero.*

Proof. If $\text{concave}(r)$ is zero, then by definition r is on the boundary of the convex hull of P, ∂H_P. If r is a vertex of ∂H_P, then the dihedral angle of r must be less than 180°. If r is on an edge of ∂H_P, then the dihedral angle of r is 180°. In both cases, r is not a notch. □

Note that the other direction of Lemma 6.1 will not be true. A non-notch vertex x may have concavity larger than zero if x is inside a pocket.

Lemma 6.2. *Assume concavity is measured using one of the methods we have proposed (SL, SP, H1 or H2). Let $x \in \partial P$ be a point with maximum concavity, i.e., $\nexists y \in \partial P$ such that* $\text{dist}(y, H_P) > \text{dist}(x, H_P)$. *Then x must be* *a notch.*

Proof. We prove this Lemma by defining properties of a set of retraction functions in which x is assumed be a notch and then we show that our proposed concavity measure functions have these properties.

We note that internal co-linear vertices do not contribute to the shape of P. Therefore, without loss of generality, all our algorithms and analysis assume such vertices do not exist (they can easily be removed in pre-processing), and hence we are guaranteed that no two vertices on ∂P will have the same concavity.

Let pocket polygon P_ρ be a polygon enclosed by a bridge β and a pocket ρ of P and let V_ρ be the vertices of P_ρ. A retraction function γ of P_ρ maps every point x in P_ρ to a point in β. The concavity of x defined over γ is $\text{concave}_\gamma(x) = \int_0^1 \gamma(x, \beta, t)\, \mathrm{d}t$. For simplicity, we denote $\gamma(x)$ as the retracting trajectory of x. Let $P_\rho^0 = P_\rho$ and let $V_\rho^0 = V_\rho$. Let V_ρ^{i+1} denote the vertices that remain after all notches are deleted from V_ρ^i and let P_ρ^i be the polygon defined by V_ρ^i. We say γ is *simple* if:

$$\text{concave}_\gamma(P_\rho^i) > \text{concave}_\gamma(P_\rho^j),\ \forall i < j, \tag{6}$$

where $\text{concave}_\gamma(P_\rho^k) = \max_{x \in V_\rho^k}\{concave_\gamma(x)\}$, and we say γ is *static* if:

$$\gamma(x) \text{ in } P_\rho^i \text{ equals } \gamma(x) \text{ in } P_\rho^j \text{ if } x \in P_\rho^i \text{ and } x \in P_\rho^j,\ \ i \neq j \tag{7}$$

Hence, if γ is static, then deleting notches from V_ρ^i will not affect the concavity of the remaining vertices, i.e., vertices in V_ρ^{i+1}. Therefore, when γ is static and simple, the concavity of P_ρ^{i+1} is decreased because the vertex x with the maximum concavity in P_ρ^i is deleted. Thus, x must be in $V_\rho^i \setminus V_\rho^{i+1}$ and x must be a notch.

These properties lead us to define a *retraction* function γ as a function that is both simple and static. We now show that SL-concavity and SP-concavity and our method for measuring the hole concavity are both simple and static. We first consider SL-concavity. Assume β is aligned along the x-axis. SL-concavity is static because vertices are always retracted in the direction of the y-axis. Let x be the lowest vertex on the y-axis. Since all vertices are above x, x cannot have an internal angle less than 180°, i.e., x must be a notch. Therefore, SL-concavity must also be simple. We next consider SP-concavity. Since all end points of the visibility tree are notches, deleting notches must reduce the concavity and will not affect the concavity of the remaining vertices. Thus, SP-concavity is simple and static. For hole concavity, if we assume β is perpendicular to the PA, then it is not difficult to see that hole concavity is similar to SL-concavity with the PA serving as the y-axis (i.e., the maximum concavity of a hole is the distance between the antipodal pair along the PA). Hence, hole concavity is also simple and static. $\square$

Although Algorithm 4.1 does not look for notches explicitly, Lemma 6.2 establishes that Algorithm 4.1 indeed resolves notches and *only* notches. Note that, although we only discuss a few concavity measures in this paper, our framework will work correctly as long as the retraction function is both simple and static.

In Lemma 6.3, we show that Algorithm 4.1 resolves *all* notches when the tolerable concavity is zero. In this case, the approximate convex decomposition is an exact convex decomposition, i.e., $\mathrm{CD}_\tau(P)$ is equal to $\mathrm{CD}(P)$.

Lemma 6.3. *Polygon P is* 0-approximate *convex if and only if P is convex.*

Proof. If P is convex, then P has no notches. In this case, the concavity of P is $\max_{x\in P}\{\mathrm{concave}(x)\} = \max_{x\in\partial P}\{\emptyset\} = 0$. Assume P is not convex but that it has zero concavity. Since P is not convex, P has at least one notch $r \neq 0$. From Lemma 6.1, we know that $\mathrm{concave}(r) \neq 0$ and thus also $\mathrm{concave}(P) \neq 0$. This contradiction establishes the lemma. $\square$

Based on Lemma 6.2 and Lemma 6.3, we conclude our analysis of Algorithm 4.1 in Theorems 6.4 and 6.5.

Theorem 6.4. *When $\tau = 0$, Algorithm 4.1 resolves* **all** *and* **only** *notches of polygon P using the concavity measurements in Section 5.*

Theorem 6.5. *Let $\{C_i\}$, $i = 1, \ldots, m$, be the τ-approximate convex decomposition of a polygon P with n vertices, r notches and k holes. P can be decomposed into $\{C_i\}$ in $O(nr)$ time.*

Proof. We first consider the case in which P has no holes, i.e., $k = 0$. We will show that each iteration in Algorithm 4.1 takes $O(n)$ time. For each iteration, we compute the convex hull of P and the concavity of P. The convex hull of P can be constructed in linear time in the number vertices of P [36]. To compute the concavity of P, we need to find bridges and pockets and compute the distance from the pockets to the bridges. Associating the bridges and pockets requires $O(n)$ time using a traversal of the vertices of P. When the shortest path distance is used, measuring $\mathrm{concave}(P)$ takes linear time as shown in Lemma 5.4. When the straight line distance is used, each measurement of concave(x) takes constant time, where x is a vertex of P. Therefore, the total time for measuring concave(P) takes $O(n)$ as well. Similarly, we can show that the hybrid approach takes $O(n)$ time. Moreover, **Resolve** splits P into C_1 and C_2 in $O(n)$ time. Thus, each iteration takes $O(n)$ time for P when P does not have holes.

If the resulting decomposition has m components, the total number of iterations of Algorithm 4.1 is $m - 1$. Since each time we split P into C_1 and C_2, at most three new vertices are created, the total time required for the $m-1$ cuts is $O(n + (n+3) + \ldots + (n + 3 * (m-2))) = O(nm + 3 \times \frac{(m-1)^2}{2}) = O(nm + m^2)$.

When $k > 0$, we estimate the concavity of a hole locally using its principal axis ($O(n)$ time) and add a diagonal between the vertex with the maximum estimated concavity and its closest vertex of ∂P ($O(n)$ time). For each hole that connects to ∂P, at most three new vertices are created. Therefore, resolving k holes takes $O(nk + k^2)$ time.

Therefore, the total time required to decompose P into $\{C_i\}$ is $O(nm+m^2)+O(nk+k^2) = O(n(m+k)+m^2+k^2)$ time. Since $m \leq r+1$ and $k < r$, $O(n(m+k)+m^2+k^2) = O(nr+r^2)$. Also, because $r < n$, $O(nr+r^2) = O(nr)$. Thus, decomposition takes $O(nr)$ time. □

The number of components in the final decomposition, m, depends on the tolerance τ and the shape of the input polygon P. A small τ and an irregular boundary will increase m. However, m must be less than $r+1$, the number of notches in P, which, in turn, is less than $\lfloor \frac{n-1}{2} \rfloor$. Detailed models, such as the Nazca line monkey and heron in Figures 1 and 16, respectively, generally have r close to $\Theta(n)$. In this case, Chazelle and Dobkin's approach [12] has $O(n+r^3) = O(n^3)$ time complexity and Keil and Snoeyink's approach [27] has $O(n+r^2 \min\{r^2, n\}) = O(n^3)$ time complexity. When $r = \Theta(n)$, Algorithm 4.1 has $O(n^2)$ time complexity.

7 Experimental Results

7.1 Models

The polygons used in the experiments are shown in Figures 14–17. The models in Figures 14–16 have no holes and the model in Figure 17 has 18 holes. The models in Figure 15 and 16 are referred to as *monkey*$_1$ and *heron*$_1$, respectively. Two additional polygons, with the same size and shape as $monkey_1$ and $heron_1$, are called $monkey_2$ and $heron_2$. Summary information for these models is shown in Table 2.

7.2 Implementation Details

We implement the proposed algorithm in C++, and use FIST [21] as the triangulation subroutine for finding the shortest paths in pockets. Instead of resolving a notch r using a diagonal that bisects the dihedral angle of r, we use a heuristic approach intended to appeal to human perception. When selecting the diagonal for a particular notch r, we consider all possible diagonals $f(r,x)$ from r to a boundary point $x \in \partial P_0$. All diagonals are scored using the following equation and the highest scoring one is selected as the diagonal for resolving r.

$$f(r,x) = \begin{cases} 0 & : \overline{rx} \text{ does not resolve r} \\ \frac{(1+s_c \times \text{concave}(x))}{(s_d \times \text{dist}(r,x))} & : \text{otherwise, where } \mathrm{s_c} \text{ and } \mathrm{s_d} \text{ are user defined scalars} \end{cases} \tag{8}$$

According to experimental studies [40], people prefer short diagonals to long diagonals. Thus, in addition to the concavity, we consider the distance as another criterion when selecting the diagonal to resolve r. Increasing s_c favors concavity and increasing s_d places more emphasis on the distance criterion. In our experiments, $s_c = 0.1$ and $s_d = 1$ are used. This scoring process adds $O(n)$ time to each iteration and therefore does not change the overall asymptotic bound.

7.3 Experimental Results

All experiments were done on a Pentium 4 2.8 GHz CPU with 512 MB RAM. They were designed to compare the final decomposition size and the execution time of the approximate convex decomposition (ACD) computed using different concavity measures and with the minimum component exact convex decomposition (MCD) [27]. For a fair comparison, we re-coded the MCD implementation available at [42] from Java to C++. To provide an additional metric for comparison, we estimate the *quality* of the final decomposition $\{C_i\}$ by measuring its *convexity* [46]:

$$convex(\{C_i\}) = \frac{\sum_i \text{area}(C_i)}{\sum_i \text{area}(H_{C_i})} , \tag{9}$$

where area(x) is the area of an object x and H_x is its convex hull. Eq. 9 provides a *normalized* measure of the similarity of the $\{C_i\}$ to their convex hulls. Thus, unlike our concavity measurements, this convexity measurement is independent of the size, i.e., area, of polygons. For example, a set of convex objects will have convexity 1 regardless of their size.

A general observation from our experiments is that when a little non-convexity can be tolerated, the ACD may have significantly fewer components and it may be computed significantly faster; see Table 3.

The ACD also generates visually meaningful components, such as legs and fingers of the monkey in Figure 1 and wings and tails of the heron in Figure 16. More results that demonstrate this property are shown in Figures 18 to 21.

Finally, when exact convex decomposition is needed ($\tau = 0$), our method does produce somewhat more components than the MCD, but it is also noticeably faster.

The maze-like model (Figure 14) illustrates differences among the concavity measures. When $\tau > 10$, the convexity measurements in Figure 14(d) show that SL-concavity misses some important features that are found by SP-concavity (and thus also by H1-concavity and H2-concavity). We also see that SP-concavity is more expensive to compute and that H2-concavity is "shape" sensitive, i.e., H2-concavity requires more (less) time if the input shape is complex (simple). Computing H2-concavity is also faster than computing H1-concavity.

The results for the larger monkey and heron models (Figures 15 and 16) show that significant savings can be obtained from ACDs with 'almost' convex components. For example, for the monkey, the radius of its bounding circle is about 82, and so 0.1 concavity means a one pixel dent in an 820×820 image, which is almost unnoticeable to bare eye. Moreover, the convexity of 0.1-convex components of $monkey_1$ ($monkey_2$) is 0.997 (0.995) and the convexity of 0.1-convex components of $heron_1$ ($heron_2$) is 0.98 (0.976). No MCD data is collected for $monkey_2$ and $heron_2$ due to the difficulty of solving these large problems with the MCD code.

This experiment reveals another interesting property of the ACD: regardless of the complexity of the input, the ACD generates almost identical decompositions for models with the same shape when τ is above a certain value. For example, when $\tau > 0.01$, ACD generates the same number of components for both $monkey_1$ and $monkey_2$ and for $heron_1$ and $heron_2$.

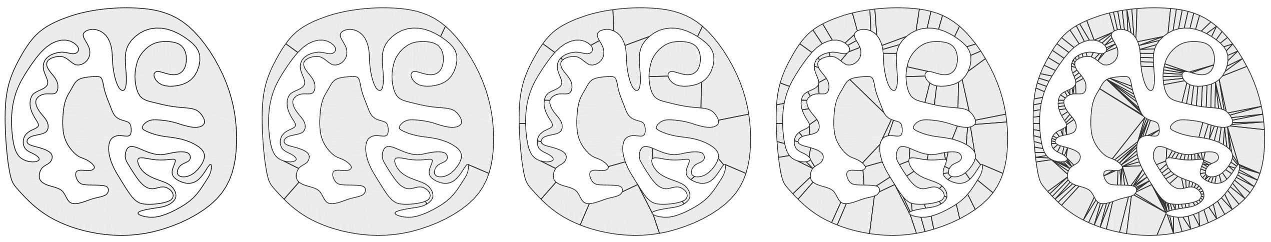
A polygonal model of planar neuron contours is shown in Figure 17. It has 18 holes and roughly 45% of the vertices are on hole boundaries. Figure 17(b) shows the decomposition using the proposed hole concavity and SP-concavity measures. The dashed line (at $Y = 0.06$) in Figure 17(c) is the total time for resolving the 18 holes. Once all holes are resolved, the ACD produces similar results as before. No MCD was computed because the algorithm cannot handle holes.

8 Conclusion

We proposed a method for decomposing a polygon into approximately convex components that are within a user-specified tolerance of convex. When the tolerance is set to zero, our method produces an exact convex decomposition in $O(nr)$ time which is faster than existing $O(nr^2)$ methods that produce a minimum number of components, where n and r are the number of vertices and notches, respectively, in the polygon. We proposed some heuristic measures to approximate our intuitive concept of concavity: a fast and inaccurate straight line (SL) concavity, a slower and more precise shortest path (SP) concavity, and hybrid (H1 and H2) concavity methods with some of the advantages of both. We illustrated that our approximate method can generate substantially fewer components than an exact method in less time, and in many cases, producing components that are τ-approximately convex. Our approach was seen to generate visually meaningful components, such as the legs and fingers of the monkey in Figure 1 and the wings and tail of the heron in Figure 16.

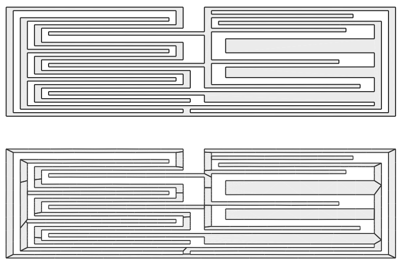
An important feature of our approach is that it also applies to polygons with holes, which are not handled by previous methods. Our method estimates the concavities for points in a hole locally by computing the "diameter" of the hole before the hole boundary is merged into the external boundary.

One criterion of the decomposition is to minimize the concavity of its components. Our decomposition method does not try to find a cut that splits a given polygon P into two components with minimum concavity. There are two reasons that we do not do so. First, greedily minimizing concavity does not necessarily produce fewer components. Second, the decomposed components with minimum concavity may not represent significant features. For instance, in order to minimize the convexity of P in Figure 22(a), P will be decomposed into P_1 and P_2 so that $\max(\text{concave}(P_1), \text{concave}(P_2))$ is minimized. However, doing so splits the polygon at unnatural places and

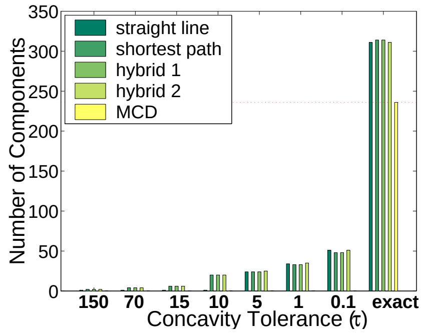


(original) ($\tau = 5$) ($\tau = 1$) ($\tau = 0.1$) ($\tau = 0$)

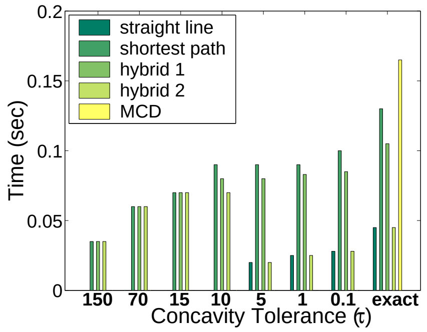
Figure 13: The original polygon has 816 vertices and 371 notches and three holes. The radius of the bounding circle is 8.14. When $\tau = 5$, 1, 0.1, and 0 units there are 4, 22, 88, and 320 components.



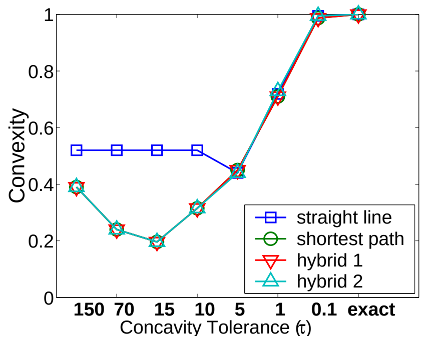
(a)



(b)

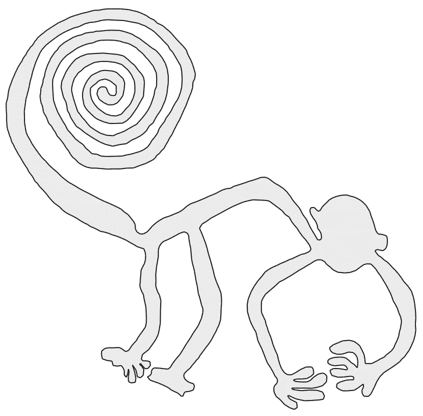


(c)

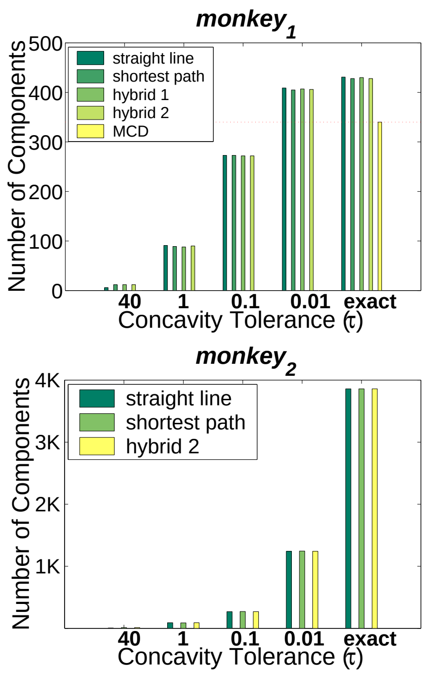


(d)

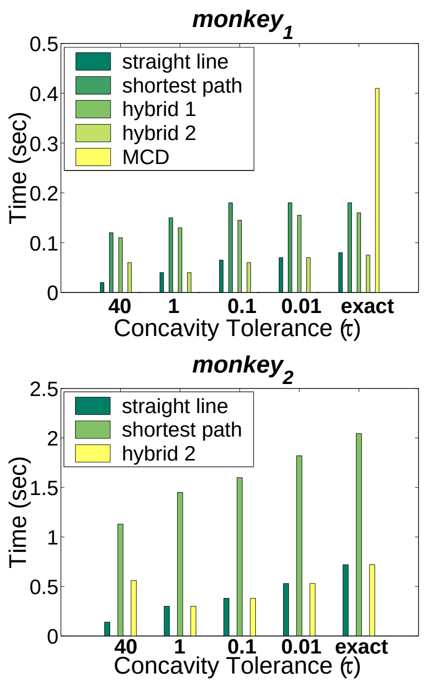
Figure 14: (a) Initial (top) and approximately (bottom) decomposed Maze model. Initial Maze model has 800 vertices and 400 notches. (b) Number of components in final decomposition. (c) Decomposition time. (d) Convexity measurements.



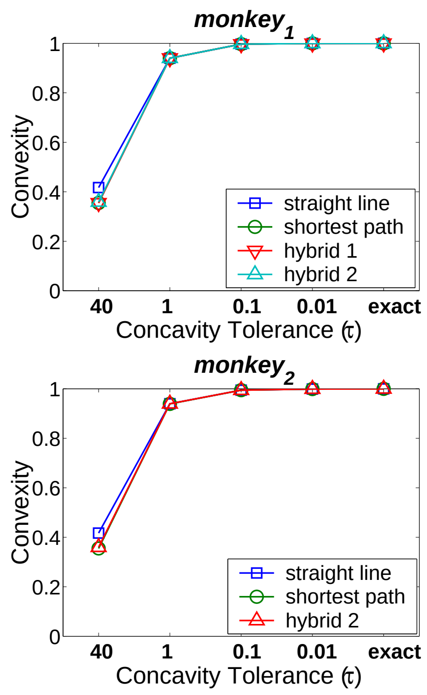
(a)



(b)

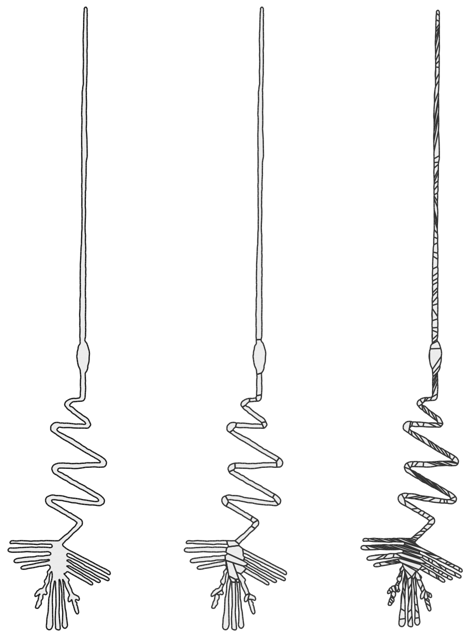


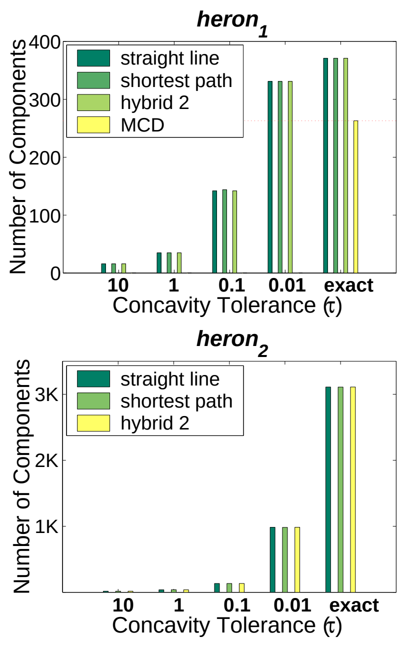
(c)



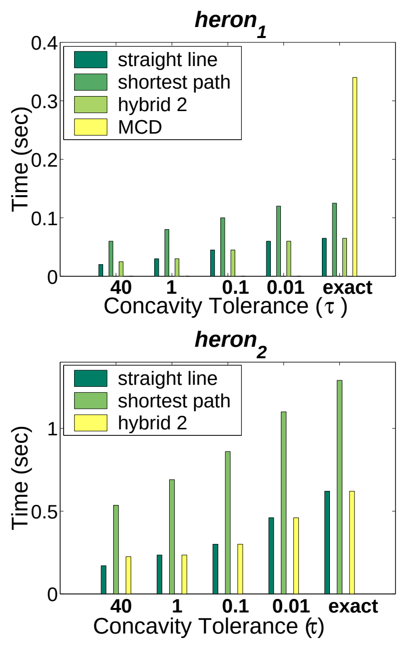
(d)

Figure 15: (a) Initial model of Nazca Monkey; see Figure 1. (b) Number of components in final decomposition. **Top:** *monkey*$_1$. **Bottom:** *monkey*$_2$. (c) Decomposition Time. (d) Convexity measurements.

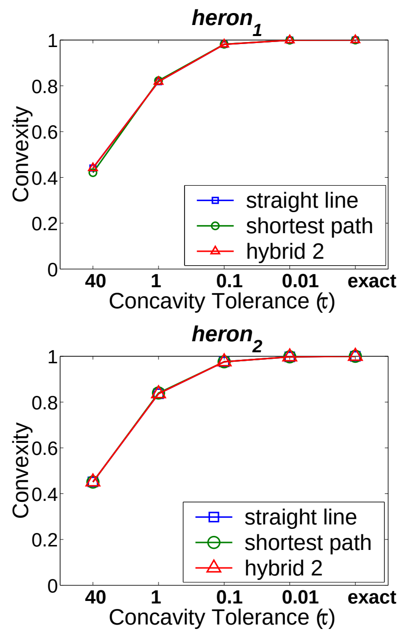

(a)



(b)

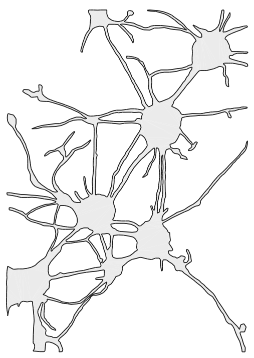


(c)

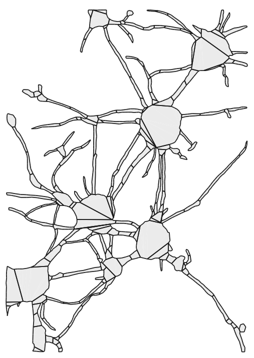


(d)

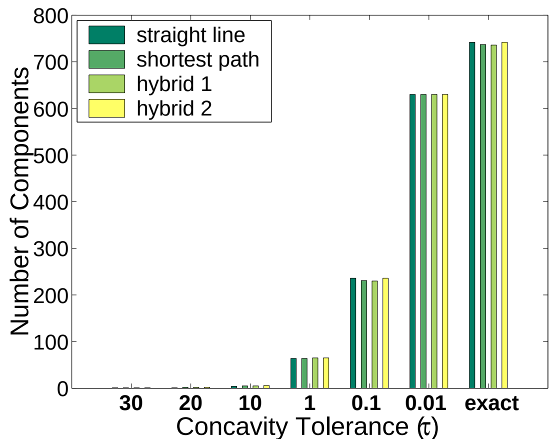
Figure 16: (a) **Top:** The initial Nazca Heron model has 1037 vertices and 484 notches. The radius of the bounding circle is 137.1 units. **Middle:** Decomposition using approximate convex decomposition. 49 components with concavity less than 0.5 units are generated. **Bottom:** Decomposition using optimal convex decomposition. 263 components are generated. (b) Number of components in final decomposition. **Top:** $heron_1$. **Bottom:** $heron_2$. (c) Decomposition time. (d) Convexity measurements.



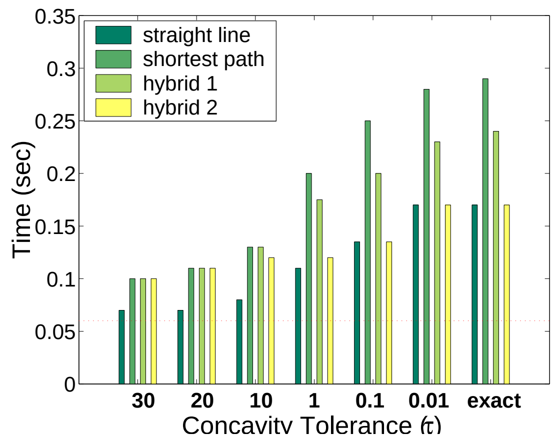
(a)



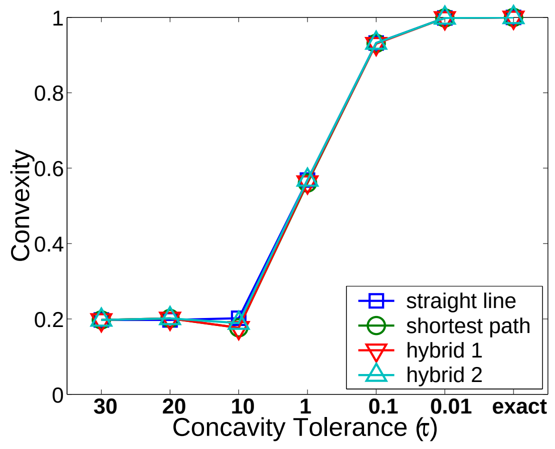
(b)



(c)



(d)



(e)

Figure 17: (a) The initial model of neurons has 1,815 vertices and 991 notches and 18 holes. The radius of the enclosing circle is 19.6 units. (b) Decomposition using approximate convex decomposition. Final decomposition has 236 components with concavity less than 0.1 units. (c) Number of components in final decomposition. (d) Decomposition Time. The dashed line indicates the time for resolving all holes. (e) Convexity measurements.

will ultimately generate more components.

While there is an increasing need for methods to decompose 3D models due to hardware advances that facilitate the generation of massive models, this problem is far less understood than its 2D counterpart. One attractive feature of the 2D approximate convex decomposition approach presented here is that it extends naturally to 3D [31], and we are developing a method based on it for extracting 3D skeletons [30]. Another possible extension is to use the concavity measurements proposed in this paper as alternative shape descriptors.

References

[1] P. K. Agarwal, E. Flato, and D. Halperin. Polygon decomposition for efficient construction of minkowski sums. In *European Symposium on Algorithms*, pages 20–31, 2000.

[2] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Linear-time polygon triangulation made easy via randomization. In *Proceedings of the 16th Annual ACM Symposium on Computational Geometry (SoCG'00)*, pages 201–212, 2000. Invited submission to special issue of *Discrete and Computational Geometry* featuring selected papers from the ACM Symposium on Computational Geometry (SoCG 2000).

[3] D. Avis and G. T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Trans. Comput.*, C-30(12):910–1014, 1981.

[4] O. E. Badawy and M. Kamel. Shape representation using concavity graphs. *ICPR*, 3:461–464, 2002.

[5] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[6] G. Borgefors and G. S. di Baja. Analyzing nonconvex 2d and 3d patterns. *Computer Vision and Image Understanding*, 63(1):145–157, 1996.

[7] G. Borgefors and G. Sanniti di Baja. Methods for hierarchical analysis of concavities. In *Proceedings of the Conference on Pattern Recognition (ICPR)*, volume 3, pages 171–175, 1992.

[8] G. Castillero. Ancient, giant images found carved into peru desert, October 2002. National Geographic News. http://news.nationalgeographic.com/news/2002/10/1008_021008_wire_peruglyphs.html.

[9] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 339–349, 1982.

[10] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[11] B. Chazelle and D. P. Dobkin. Decomposing a polygon into its convex parts. In *Proc. 11th Annu. ACM Sympos. Theory Comput.*, pages 38–48, 1979.

[12] B. Chazelle and D. P. Dobkin. Optimal convex decompositions. In G. T. Toussaint, editor, *Computational Geometry*, pages 63–133. North-Holland, Amsterdam, Netherlands, 1985.

[13] F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete and Computational Geometry*, 21(3):405–420, 1999.

[14] A. G. Cohn. A hierarchical representation of qualitative shape based on connection and convexity. In *International Conference on Spatial Information Theory*, pages 311–326, 1995.

[15] E. D. Demaine, M. L. Demaine, and J. S. B. Mitchell. Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami. In *Symposium on Computational Geometry*, pages 105–114, 1999.

[16] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In *Proc. Workshop on Algorithms and Data Structures*, pages 25–36, 2003.

[17] H. Y. F. Feng and T. Pavlidis. Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition. *IEEE Trans. Comput.*, C-24:636–650, 1975.

[18] T. Fevens, H. Meijer, and D. Rappaport. Minimum convex partition of a constrained point set. In *Abstracts 14th European Workshop Comput. Geom.*, pages 79–81. Universitat Polytènica de Catalunya, Barcelona, 1998.

[19] D. H. Greene. The decomposition of polygons into convex parts. In F. P. Preparata, editor, *Computational Geometry*, volume 1 of *Adv. Comput. Res.*, pages 235–259. JAI Press, Greenwich, Conn., 1983.

[20] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[21] M. Held. FIST: Fast industrial-strength triangulation of polygons. Technical report, University at Stony Brook, 1998.

[22] S. Hert and V. J. Lumelsky. Polygon area decomposition for multiple-robot workspace division. *International Journal of Computational Geometry and Applications*, 8(4):437–466, 1998.

[23] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961, 2003.

[24] J. M. Keil. *Decomposing Polygons into Simpler Components*. PhD thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, 1983.

[25] J. M. Keil. Decomposing a polygon into simpler components. *SIAM J. Comput.*, 14:799–817, 1985.

[26] J. M. Keil. Polygon decomposition. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 491–518. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[27] M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. In M. Soss, editor, *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 54–55, Montréal, Québec, Canada, 1998. School of Computer Science, McGill University.

[28] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.

[29] C. Levcopoulos and A. Lingas. Bounds on the length of convex partitions of polygons. In *Proc. 4th Conf. Found. Softw. Tech. Theoret. Comput. Sci.*, volume 181 of *Lecture Notes Comput. Sci.*, pages 279–295. Springer-Verlag, 1984.

[30] J.-M. Lien and N. M. Amato. Approximate convex decomposition. Technical Report TR03-001, Parasol Lab, Dept. of Computer Science, Texas A&M University, 2003.

[31] J.-M. Lien and N. M. Amato. Approximate convex decomposition. In *Proc. 20th Annual ACM Symp. Computat. Geom. (SoCG)*, pages 457–458, June 2004. Video Abstract.

[32] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polygons. In *Proc. 20th Annual ACM Symp. Computat. Geom. (SoCG)*, pages 17–26, June 2004.

[33] A. Lingas. The power of non-rectilinear holes. In *Proc. 9th Internat. Colloq. Automata Lang. Program.*, volume 140 of *Lecture Notes Comput. Sci.*, pages 369–383. Springer-Verlag, 1982.

[34] A. Lingas, R. Pinter, R. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proc. 20th Allerton Conf. Commun. Control Comput.*, pages 53–63, 1982.

[35] D. Marr. Analysis of occluding contour. In *Proc. Roy. Soc. London*, pages 441–475, 1977.

[36] D. McCallum and D. Avis. A linear algorithm for finding the convex hull of a simple polygon. *Inform. Process. Lett.*, 9:201–206, 1979.

[37] P. L. Rosin. Shape partitioning by convexity. *IEEE Transactions on system, man, and cybernetics - Part A : Sytem and Humans*, 30(2):202–210, March 2000.

[38] K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995.

[39] M. Simmons and C. H. Séquin. 2d shape decomposition and the automatic generation of hierarchical representations. *International Journal of Shape Modeling*, (4):63–78, 1998.

[40] M. Singh, G. Seyranian, and D. Hoffma. Parsing silhouettes: The short-cut rule. *Perception & Psychophysics*, 61:636–660, 1999.

[41] J. Sklansky. Measuring concavity on rectangular mosaic. *IEEE Trans. Comput.*, C-21:1355–1364, 1972.

[42] J. Snoeyink. Minimum convex decomposition. Available at http://www.cs.ubc.ca/∼snoeyink/demos/convdecomp/.

[43] H. I. Stern. Polygonal entropy: A convexity measure. *Pattern Recognition Letters*, 10:229–235, 1989.

[44] S. Tor and A. Middleditch. Convex decomposition of simple polygons. *ACM Transactions on Graphics*, 3(4):244–265, 1984.

[45] M. Tănase and R. C. Veltkamp. Polygon decomposition based on the straight line skeleton. In *Proceedings of the nineteenth conference on Computational geometry (SoCG)*, pages 58–67. ACM Press, 2003.

[46] J. Zunic and P. L. Rosin. A convexity measurement for polygons. In *British Machine Vision Conference*, pages 173–182, 2002.

Table 1: Nazca monkey (Figure 1(a)) decomposition using SL-, SP-, H1-, and H2-Concavity with τ as 40, 20, 10, and 1 units.

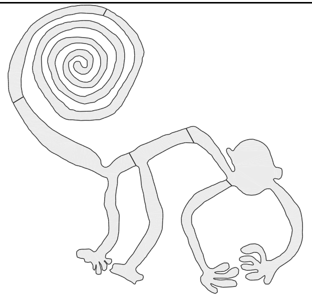
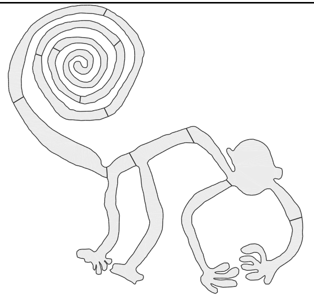
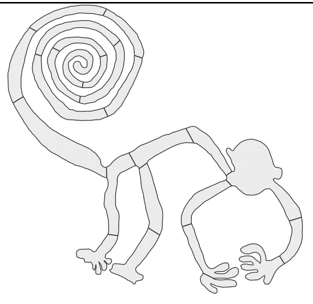
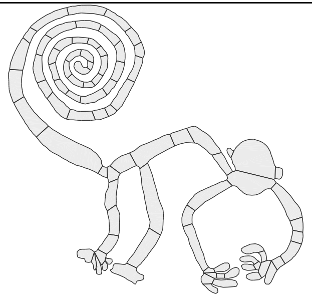
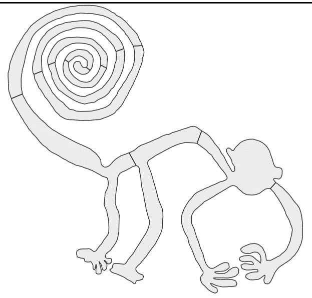
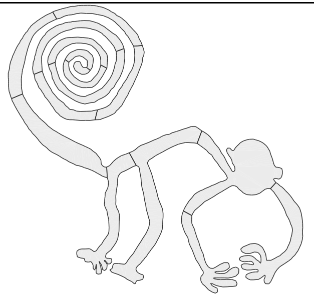
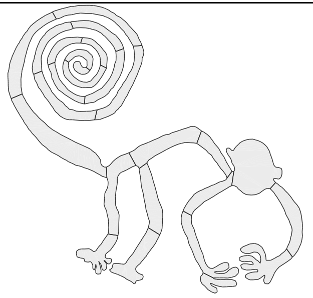
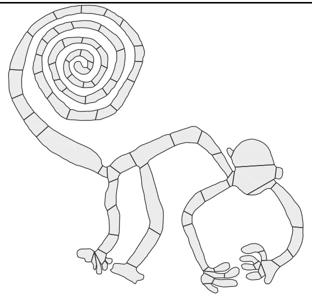
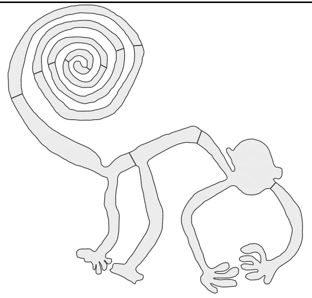
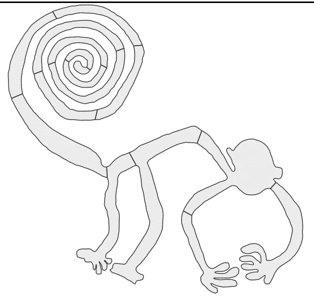
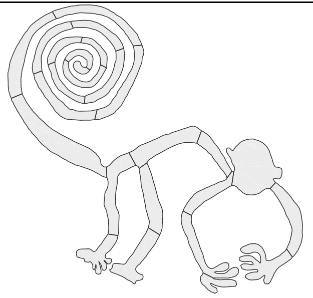
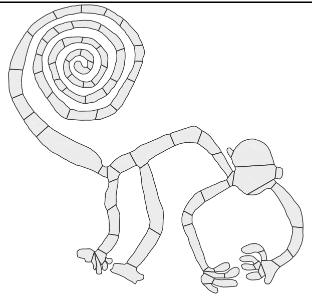
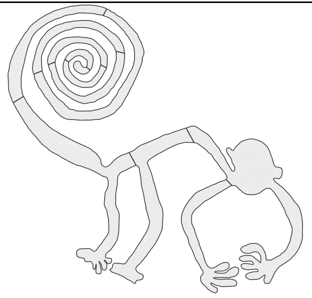
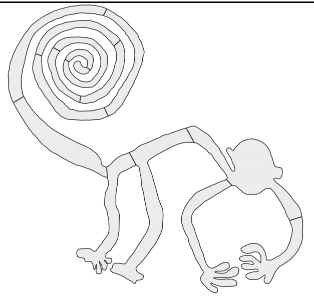
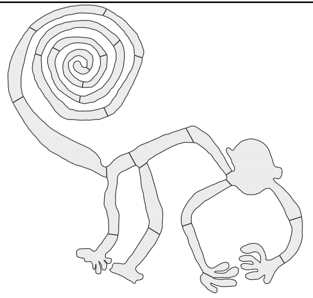
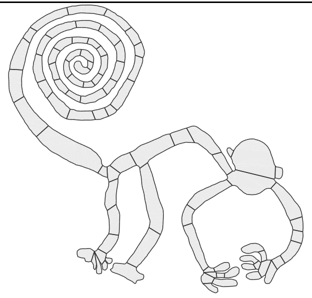
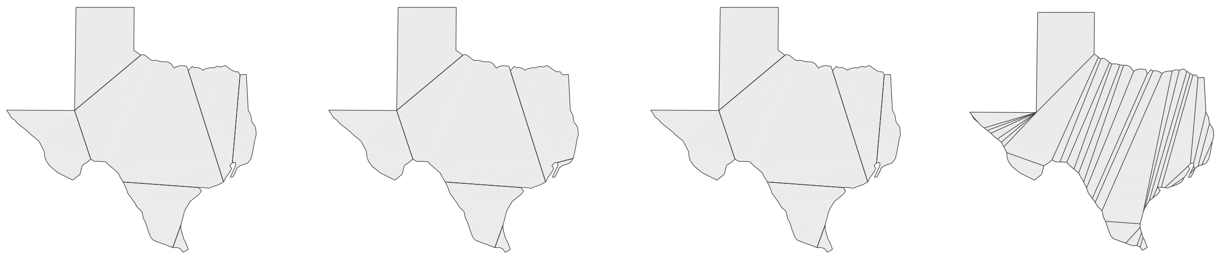
$\tau = 40$	$\tau = 20$	$\tau = 10$	$\tau = 1$
SL-Concavity			
 (6 components)	 (13 components)	 (24 components)	 (90 components)
SP-Concavity			
 (12 components)	 (16 components)	 (26 components)	 (88 components)
H1-Concavity			
 (12 components)	 (16 components)	 (26 components)	 (88 components)
H2-Concavity			
 (12 components)	 (15 components)	 (25 components)	 (90 components)

Table 2: Summary Information for Models Studied. R is the radius of the minimum enclosing ball.

Name	**# vertices**	**# notches**	**# holes**	R **(units)**
maze (Figure 14)	800	400	0	15.3
monkey$_1$ (Figure 15)	1204	577	0	81.7
monkey$_2$	9632	4787	0	81.7
heron$_1$ (Figure 16)	1037	484	0	137.1
heron$_2$	8296	4122	0	137.1
neuron (Figure 17)	1815	991	18	19.6

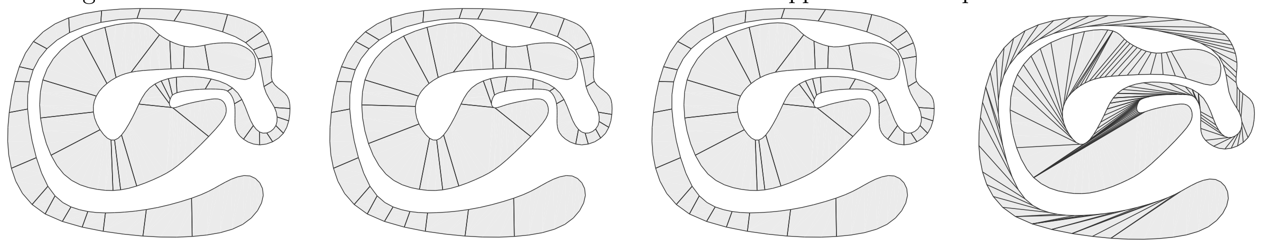
Table 3: Comparing the decomposition size and time of the ACD and the MCD. Convexity and concavity in this table indicate the tolerance of the ACD.

Name	**convexity** (unitless)	**concavity** (units)	**size** (ACD:MCD)	**time** (ACD:MCD)
maze (Figure 14)	99.5%	0.1	1:4	1:8
monkey$_1$ (Figure 15)	99.7%	0.1	8:10	1:6.3
heron$_1$ (Figure 16)	98.0%	0.1	1:2	1:7.6



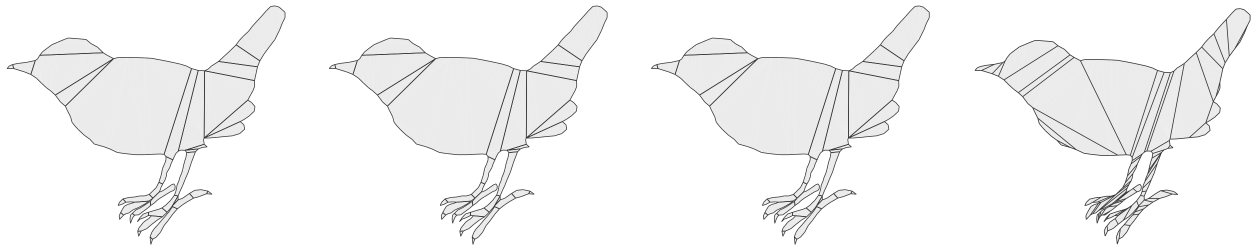
(SL, 7 components) (SP, 7 components) (H2, 7 components) (MCD, 38 components)

Figure 18: Texas. 139 vertices. 62 notches. Radius is 17.4 units. Approximate components are 1-convex.



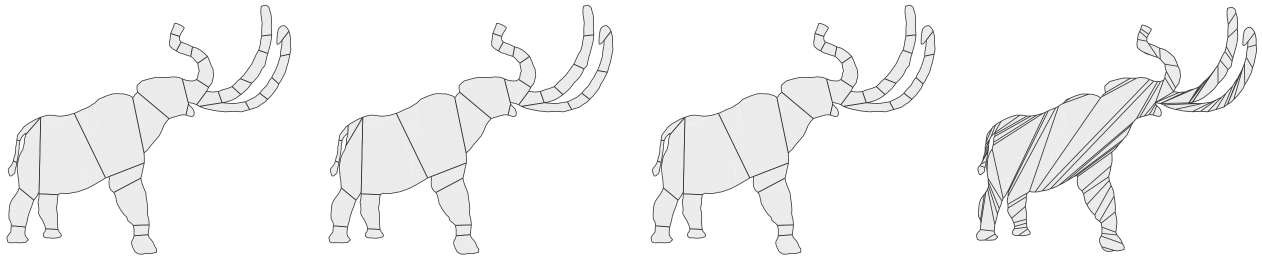
(SL, 49 components) (SP, 48 components) (H2, 49 components) (MCD, 126 components)

Figure 19: No name. 348 vertices. 153 notches. Radius is 12.9 units. Approximate components are 0.1-convex.



(SL, 37 components) (SP, 37 components) (H2, 37 components) (MCD, 75 components)

Figure 20: Bird. 275 vertices. 133 notches. Radius is 15.4 units. Approximate components are 0.1-convex.



(SL, 35 components) (SP, 34 components) (H2, 34 components) (MCD, 105 components)

Figure 21: Mammoth. 403 vertices. 185 notches. Radius is 16.5 units. Approximate components are 0.2-convex.

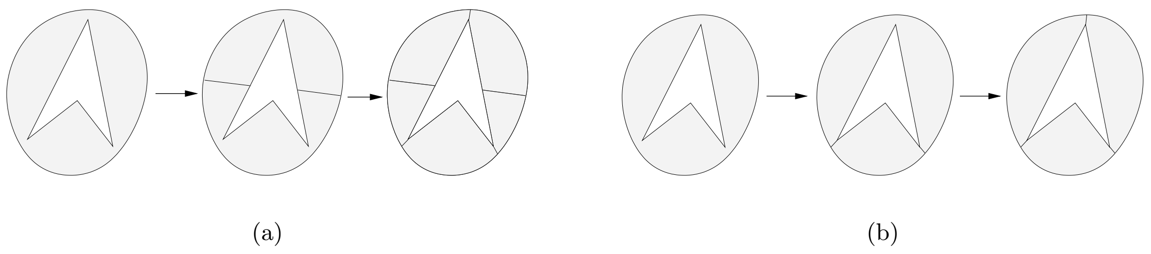


Figure 22: (a) Decomposition that minimizes concavity. (b) Decomposition using the proposed method.