Approximate Convex Decomposition of Polygons[*]

Jyh-Ming Lien
Parasol Lab, Dept. of Computer Science
Texas A&M University
neilien@cs.tamu.edu

Nancy M. Amato
Parasol Lab, Dept. of Computer Science
Texas A&M University
amato@cs.tamu.edu

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; a user specified tolerance determines allowable concavity. 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. 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.

Categories and Subject Descriptors: I.3.5 [Computing Methodologies]: Computer Graphics—*Computational Geometry and Object Modeling*

General Terms: Algorithms, Theory

Keywords: convex decomposition, hierarchical, polygon.

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 [11], Minkoski sum computation [1], motion planning [15], computer graphics [20], and origami folding [9].

[*]This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0081510, ACR-0113971, CCR-0113974, EIA-9810937, EIA-0079874, and by Texas HECB grant ATP-000512-0261-2001.

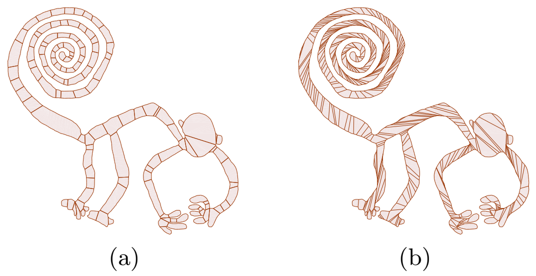


Figure 1. (a) Approximate ($\tau = 0.5$) and (b) exact minimum convex decompositions using no Steiner points of a Nazca monkey (Figure 16(a)), with 128 and 340 components, respectively.

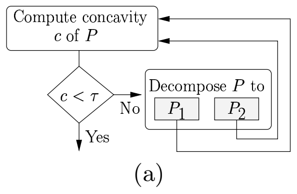
One issue with convex decompositions, however, is that they can be costly to construct and can result in representations with an unmanageable number of components. For example, while a minimum set of convex components can be computed efficiently for simple polygons without holes [18], the problem is NP-hard for polygons with holes [22].

In this work, 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 *approximate convex decomposition* (ACD) provide similar benefits as convex components, while the ACD 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 ACD and a minimum convex decomposition [18] of a Nazca line monkey.[†]

[†]Nazca lines are mysterious drawings found in southwest Peru.



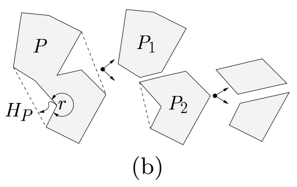


Figure 2. (a) Decomposition continues until user input concavity tolerance τ is met. (b) Hierarchical representation of polygon P; the concavity of notch r is the distance to P's convex hull H_P.

Our approach is based on the premise that for some models and applications, some of the non-convex (concave) 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 example of the decomposition process is shown in Figure 2(a). Due to the recursive application, the resulting decomposition is a hierarchical binary tree; see Figure 2(b). 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, our approach also constructs a hierarchical representation that provides multiple Levels of Detail (LOD). A single decomposition is constructed based on the highest accuracy needed, but coarser, "less convex" models 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 using an approximate decomposition, the sensitivity to small surface features can be removed, or at least decreased.

The success of our approach depends critically on the quality 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 a 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 measures 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 investigate 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. We say an ACD is τ-convex if the concavity of all vertices of all components is at most τ.

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.

2. PRELIMINARIES

A simple 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 $\partial P_{1\le i\le 4}$ and the region bounded by ∂P_5 are processed separately.

The convex hull of a polygon P, H_P, is the smallest convex set enclosing P. P is said to be convex if $P = H_P$. Vertices of P that are not vertices of H_P are *notches*, i.e., notches have internal angles greater than 180°. 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:

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

A *convex decomposition* of P is a decomposition $D(P)$ in which all components are convex, i.e.,

$$\mathrm{CD}(P) = \{C_i \mid C_i \in \mathrm{D}(P) \text{ and } C_i \text{ is convex}\}. \quad (2)$$

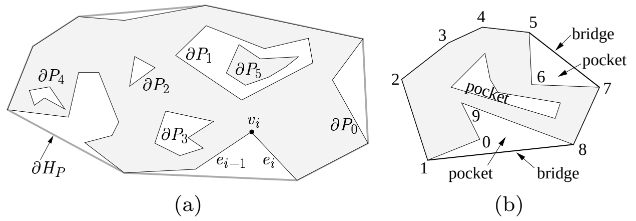


Figure 3. (a) A simple polygon with nested holes. (b) Edges (5,7) and (8,1) are bridges with associated pockets $\{(5,6),(6,7)\}$ and $\{(8,9),(9,0),(0,1)\}$, respectively.

Our concavity measures use the concepts of *bridges* and *pockets*. 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., $\mathrm{POCKETS}(P) = \partial P \setminus \partial H_P$. See Figure 3(b). Observation 2.1 states the relationship between bridges, pockets, and notches.

OBSERVATION 2.1. *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, e.g., [17]. The problem of convex decomposition of a polygon is usually subject to some optimization criteria, the most common being to produce a minimum number of convex components or to minimize the total length of the boundary of these components.

For polygons with holes, the problem is NP-hard for both the minimum components criterion [22] and the shortest internal length criterion [16, 23].

When applying the minimum component criterion for polygons without holes, the situation varies depending on whether or not Steiner points are allowed. When Steiner points are not allowed, Chazelle [6] presents an $O(n \log n)$ time algorithm that produces fewer than $4\frac{1}{3}$ times the optimal number of components. Later, Green [12] provided an $O(r^2 n^2)$ algorithm to generate the minimum number of convex components. Keil [16] improved the running time to $O(r^2 n \log n)$, and more recently Keil and Snoeyink [18] improved the time bound to $O(n + r^2 \min(r^2, n))$. When Steiner points are allowed, Chazelle and Dobkin [7] use 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. Their algorithm runs in $O(n + r^3)$ time.

When applying the shortest internal length criterion for polygons without holes, Greene [12] and Keil [16] proposed $O(r^2 n^2)$ and $O(r^2 n^2 \log n)$ 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 [19] produces a solution of length $O(p \log r)$ in time $O(n \log n)$, where p is the length of the perimeter of the polygon.

More recently, several methods have been proposed to partition at salient features of a polygon. Siddiqi and Kimia [26] use curvature and region information to identify *limbs* and *necks* of a polygon and use them to perform decomposition. Simmons and Séquin [27] proposed a decomposition using an *axial shape graph*, a weighted Medial Axis. Tănase and Veltkamp [32] decompose a polygon based on events encountered during straight-line skeleton construction that indicate the annihilation or creation of certain features. Dey et al. [10] 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) or post-processing (e.g., merging over-partitioned components) to adjust for boundary noise.

4. APPROXIMATE CONVEX DECOMPOSITION

Research in Psychology has shown that humans recognize shapes by decomposing them into components [4, 24, 26, 28]. 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.

More formally, our goal is to generate τ-*approximate* convex decompositions, where τ is a tunable parameter denoting the non-concavity tolerance of the application. For a given polygon P, P is said to be τ-*approximate* convex if concave$(P) < \tau$, where concave(ρ) denotes the concavity measurement of ρ. We discuss concavity measures in Section 5; here we assume such a measure exists. A τ-*approximate* convex decomposition of P, $\mathrm{CD}_\tau(P)$, is defined as a decomposition that contains only τ-*approximate* convex components; i.e.,

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

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

Algorithm 4.1 Approx_CD(P, τ)

```
Input. A polygon $P$ and tolerance $\tau$.
Output. A decomposition of $P$, $\{C_i\}$, such that
    $max\{\mathrm{concave}(C_i)\} \le \tau$.
1: Let a point $x \in \partial P$ be a witness for concave($P$).
2: if concave($P$) $< \tau$ then
3:    return $P$.
4: else
5:    $\{C_i\}$ = Resolve($P, x$).
6:    for Each component $C \in \{C_i\}$ do
7:       Report Approx_CD($C$,$\tau$).
```

Algorithm 4.1 describes a *divide-and-conquer* strategy to decompose P into a set of τ-*approximate* convex pieces. The algorithm first finds a point, $x \in \partial P$, which is a witness of the concavity of P, i.e., x is the most concave feature in P. Then, if the concavity of P is above the maximum tolerable value, the **Resolve**(P, x) sub-routine will remove the concave feature at x. The manner in which we measure concavity (described in Section 5) and implement **Resolve**, ensures 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 Figure 4(a) and (b). 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 partition. 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)).

The main task that still needs to be specified in Algorithm 4.1 is how to measure the concavity of a polygon. We present several alternatives in Section 5.

5. MEASURING CONCAVITY

In contrast to measures like radius, surface area, and volume, concavity does not have a well accepted definition.

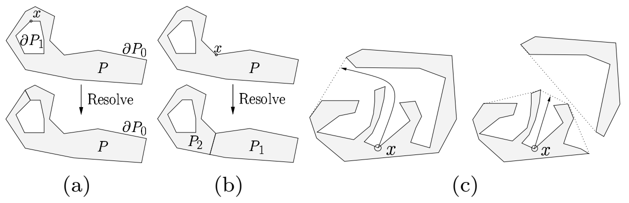


Figure 4. (a) If $x \in \partial P_{i>0}$, **Resolve** merges ∂P_i into P_0. (b) If $x \in \partial P_0$, **Resolve** splits P into P_1 and P_2. (c) The concavity of x changes after the polygon is decomposed.

For our work, however, we need a quantitative way to measure the concavity of a polygon. A few methods have been proposed [29, 5, 3] 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* [31, 33] 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 P's area to the area of P's convex hull [33], or as the probability that a fixed length line segment randomly positioned in the convex hull of P will lie entirely in P [33]. Because convexity 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 measurement. Rosin [25] 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.

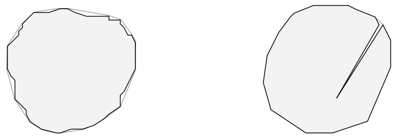


Figure 5. $\int_{\partial P_1} \text{concave}(x)\,\mathrm{d}x = \int_{\partial P_2} \text{concave}(x)\,\mathrm{d}x$, but P_1 is *visually* more convex than P_2.

Although our approach is not restricted to a particular measure, here we define the concavity of a polygon as the maximum concavity of its boundary points, i.e., $\text{concave}(P) = \max_{x\in\partial P}\{\text{concave}(x)\}$. We believe points with maximum concavity will be better identifiers of important features, than, say, summing concavities which would be similar to the convexity measurement in [31, 33]; see Figure 5.

5.1 External Boundary Concavity

An intuitive way to define concave(x) for a point $x \in P$ is to consider the trajectory of a point $x \in \partial(P)$ when x is retracted from its original position to ∂H_P. More formally, let $\text{retract}(x, H_P, t) : \partial P \to H_P$ denote the function defining this trajectory. When $t = 0$, $\text{retract}(x, H_P, 0)$ is

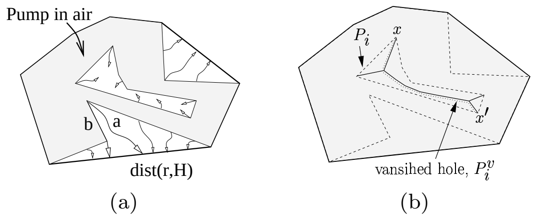


Figure 6. (a) The initial shape of a non-convex balloon (shaded). The bold line is the convex hull of the balloon. When we inflate the balloon, points not on the convex hull will be pushed toward the convex hull. Path a denotes an inflation trajectory and path b is an approximation of a. (b) The hole vanishes to a curve and vertices on the hole boundary will not touch the convex hull.

x itself. When $t = 1$, $\text{retract}(x, H_P, 1)$ is the final position of x on ∂H_P. Assuming that this retraction exists for x, $\text{concave}(x) = \text{dist}(x, H_P)$ is the integration of the function $\text{retract}(x, H_P, t)$ from $t = 0$ to 1. 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, we approximate $\text{dist}(x, H_P)$ by computing the straight-line distance from x to the bridge (this line might intersect ∂P_0). In Section 5.1.2, we use the shortest path from x to the bridge in a visibility tree computed in the pocket. A hybrid approach is 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. Figure 7 shows the decomposition of a Nazca monkey using SL-concavity.

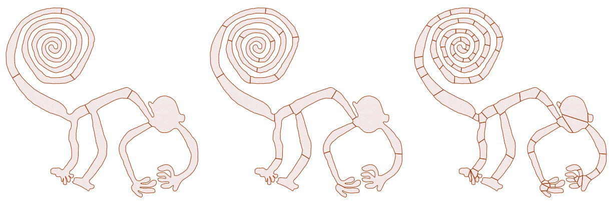


Figure 7. Nazca monkey (Figure 1(a)) decomposition using SL-Concavity. When τ is 40, 10, and 1 there are 7, 24 and 89 components, resp.

Although computing straight line distance is simple and efficient, this approach has the drawback of leaving certain types of concave features in the final decomposition. As shown in Figure 8, the concavity of s does not decrease monotonically during the decomposition. This results in the possibility of leaving important features, like s, hidden in the resulting components. This deficiency is also shown in the first image of Figure 7 ($\tau = 40$) 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.

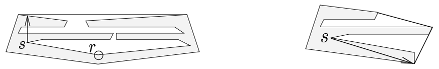


Figure 8. Let r be the notch with maximum concavity. After resolving r, the concavity of s increases. If concave(r) is less than τ, s will never be resolved even if concave(s) is larger then τ.

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 $\beta = (\beta^-, \beta^+)$ that lies entirely in the area enclosed by β and ρ, called the *pocket polygon* P_ρ. Notice that P_ρ must be a simple polygon. See Figure 9(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* [2] 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 9(a) are perpendicularly visible from the bridge β, while d is weakly visible and b is not visible. We denote by V^+_β the ordered set of vertices of P_ρ that are perpendicularly visible from β, where vertices in V^+_β have the same order as in ∂P_0.

We compute the shortest distance to β of each vertex x in ρ according to the process sketched in Algorithm 5.1. First, we split P_ρ into three regions, A, B, and C as shown in Figure 9(a). The boundaries between A and B and B and C, i.e., $\overline{a\beta^-}$ and $\overline{c\beta^+}$, are perpendicular to β. As shown in Lemma 5.2, the shortest paths for x in A or C to β are the shortest paths to β^- or β^+, respectively. These paths can be found by constructing a *visibility tree* [13] rooted at β^- (β^+) to all vertices in A (C).

The shortest path for a vertex $x \in B$ to β is composed of two parts: the path $\pi(x, y), \forall y \in V^+_\beta$, and the path $\pi(y, \beta)$. Let $V^-_\beta = \{v \in \partial B\} \setminus V^+_\beta$. Figure 9(b) illustrates an example of V^+_β and V^-_β. For each $v \in V^+_\beta$, there exists a sub-set of vertices in V^-_β that are closer to v than to $V^+_\beta \setminus v$. These vertices must have shortest paths passing through v and 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}.

We compute V^+_β using Lemma 5.1. Our problem is a degenerate case of Lemma 5.1 as one of the edges collapses into a vertex, v.

Algorithm 5.1 Dist2Bridge(β,ρ)

```
1: Split $P_\rho$ in to polygon $A$, $B$, and $C$ as shown in Figure 9(a).
2: Construct two visibility trees, $T_1$ and $T_2$, rooted at $\beta^-$ and
   $\beta^+$, respectively, to all vertices in $P_\rho$.
3: Compute $\pi(v, \beta)$, $\forall v \in A$ ($C$) from $T_1$ ($T_2$).
4: Compute an ordered set, $V^+_\beta$, in $B$ from $T_1$ and $T_2$.
5: for each pair $(v_i, v_j) \in V^+(\beta)$ do
6:    for $i < k < j$ do
7:       $\pi(v_k, \beta) = \min(\pi(v_k, v_i), \pi(v_k, v_j)) + \pi(v, \beta)$.
```

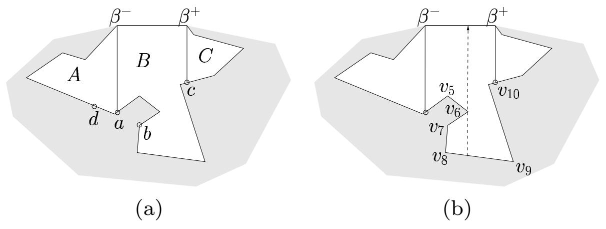


Figure 9. (a) Split P_{β^-,β^+} into A, B, and C. (b) $V^-(\beta) = \{v_7, v_8, v_9\}$ and $V^+(\beta) = \{v_5, v_6, v_{10}\}$.

Lemma 5.1. [13] *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 the bridge line segment β.

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

Proof. Straight forward; details omitted. □

The concavity of a vertex v in pocket ρ is the length of the shortest path from v to β. To compute the SP-concavity of ∂P_0, we find all bridge/pocket pairs and apply Algorithm 5.1 to each pair.

Next, we show that concave(P) decreases monotonically in Algorithm 4.1 if we use SP-concavity. The guarantee of monotonically decreasing concavity eliminates the problem of possibly leaving important concave features untreated as may happen using SL-concavity (see Figure 10).

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

Proof. New bridges constructed after a vertex in a pocket ρ is resolved must intersect the shortest paths from the other vertices in ρ to the previous bridge β. Therefore, the concavity of these vertices must decrease or remain the same. □

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

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

Proof. Straight forward; details omitted. □

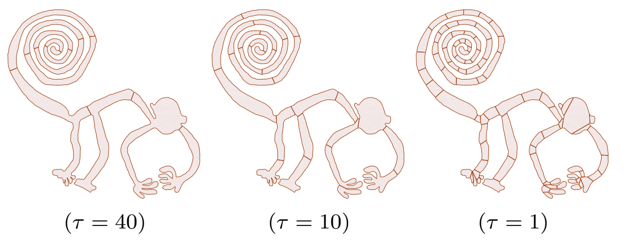


Figure 10. Nazca monkey (Figure 1(a)) decomposition using SP-Concavity. When τ is 40, 10, and 1 there are 12, 26, and 88 components, resp.

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 guarantees that concavity decreases monotonically. 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 fails to report a significant feature x when the straight-line path from x to β intersects ∂P_0. In this case, x's concavity is under measured. Such points x can be detected by comparing the directions of the outward normal $\vec{n}_\beta$ of the bridge β and the outward normals $\vec{n}_i$ for the vertices v_i in x's pocket, where $\vec{n}_i$ is the outward normal of the edge e_i incident to v_i; see Figure 11. The decision to use SL-concavity or SP-concavity is based on the following observation.

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

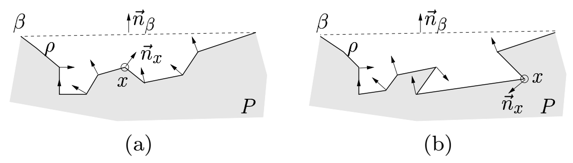


Figure 11. SL-concavity can handle the pocket in (a) correctly because none of the normal directions of the vertices in the pocket are opposite to the normal direction of the bridge. However, the pocket in (b) may result in non-monotonically decreasing concavity.

This observation leads to Algorithm 5.2. First, SL-concavity is used 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 unacceptable concavity is hidden inside the pocket. If we find a potential violation, then SP-concavity is used. Figure 12 shows the decomposition process using this hybrid measure. Experiments comparing the three approaches will be shown in Section 7.

Algorithm 5.2 Dist2Bridge(β,ρ)

```
1: Measure the SL-concavity. (Section 5.1.1)
2: **if** concave($\rho$) $> \tau$ **then**
3:     Done.
4: **if** No potential hazard detected, i.e., $\nexists r \in \rho$ such that $\vec{n}_r \cdot \vec{n}_\beta < 0$ **then**
5:     Done.
6: Measure the SP-concavity. (Section 5.1.2)
```

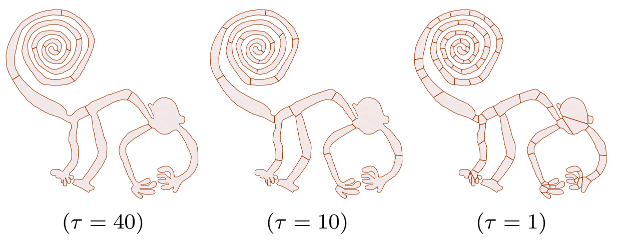


Figure 12. Nazca monkey (Figure 1(a)) decomposition using Hybrid Concavity. When τ is 40, 10, and 1 there are 12, 25, and 89 components, resp.

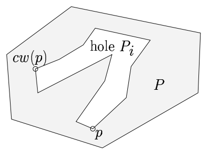
5.2 Hole Boundary Concavity

Note that in the balloon expansion analogy, points on hole boundaries will never touch H_P. The concavity of points in holes is therefore defined to be infinity and so we need some other measure for them. Our approach is to 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 as the hole 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.*

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., a diagonal between x and ∂P_0 is added, and, eventually, ∂P_i becomes 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 13. Intuitively, the maximum $\text{dist}(p, cw(p))$ represents the "diameter" of P_i, and we refer to the pair $(p, cw(p))$ realizing the maximum distance as the *antipodal pair* of the hole P_i. The antipodal pair of a hole represents an important feature 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 $cw(p)$.

To find the antipodal pair, i.e., p and $cw(p)$, of P_i, a naive approach is to exhaustively resolve all vertices in ∂P_i. Unfortunately, this approach requires $O(n_i^2)$ time, where n_i is the number of vertices of ∂P_i. Instead, we can measure the concavity of P_i locally without considering ∂P_0 and H_P. Still, computing distances between all pairs of points in ∂P_i requires $O(n_i^2)$ time.



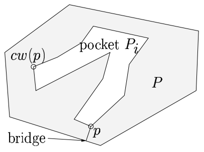


Figure 13. An example of a hole P_i and its antipodal pair. The maximum distance between p and $cw(p)$ represents the diameter of P_i. After resolving p, P_i becomes a pocket and $cw(p)$ is the most concave point in the pocket.

Fortunately, there are some possibilities to approximate p and $cw(p)$ more efficiently. As previously mentioned, the vanished hole is a very good candidate to find p and $cw(p)$ because it provides a network that 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 this network. The vanished hole can be computed using the Medial Axis Transform in $O(n)$ time [8]. Since P_i is a simple polygon, the medial axis of P_i forms a tree. 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 can be computed in $O(n)$ time.

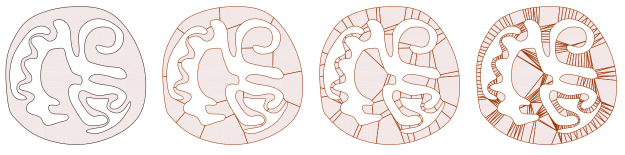
For a polygon with k holes, we compute the antipodal pair, p_i and $cw(p_i)$, for each hole P_i, $1 \leq i \leq k$. Without loss of generality, assume p_i is closer to ∂P_0 than $cw(p_i)$. A hole P_i is resolved when a diagonal is added between p_i and ∂P_0. The concavity of a hole P_i is defined to be:

$$concave(P_i) = concave(P_0) + \text{dist}(p_i, \partial P_0) + \text{dist}(p_i, cw(p_i)) \tag{5}$$

Since all vertices in a hole have infinite concavity, the term $concave(P_0)$ in Eq. 5 ensures that hole concavity is larger than the concavity of P_0, and $\text{dist}(p_i, \partial P_0) + \text{dist}(p_i, cw(p_i))$ measures how "deep" the hole is.

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 all notches must be removed (Lemma 6.3). Recall that a notch is a point on ∂P that has interior angle larger than 180°. First, observe that if x is a notch, then the concavity of x must larger than zero.



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

Figure 14. The original polygon has 816 vertices, 371 notches, and 3 holes. The radius of the bounding circle is 8.14. When $\tau = 1$, 0.1, and 0 there are 22, 88, and 320 components.

LEMMA 6.1. *If a vertex $v_i \in \partial P$ is a notch, then its concavity must be positive, i.e.,* $\text{concave}(v_i) > 0$.

PROOF. Straight forward; details omitted. □

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

LEMMA 6.2. *Assume all vertices of ∂P have distinct concavity. Let $v_i \in \partial P$ be a vertex with maximum concavity, i.e., $\nexists v_j \in \partial P$* such that $\text{dist}(v_j, H_P) > \text{dist}(v_i, H_P)$. *Then v_i must be a notch.*

PROOF. Straight forward; details omitted. □

Although Algorithm 4.1 does not look for notches explicitly, Lemma 6.2 provides the evidence that Algorithm 4.1 indeed resolves notches and *only* notches.

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., $\text{CD}_{\tau=0}(P) = \text{CD}(P)$.

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

PROOF. Straight forward; details omitted. □

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** *notches, and* **only** *the notches, of polygon P using the concavity measurements in Section 5.*

THEOREM 6.5. *Algorithm 4.1 computes a τ-approximate convex decomposition, $\{C_i\}$, $i = 1, \ldots, m$, of a simple polygon P containing an arbitrary number of holes in $O(nr^*) = O(nr)$ time, where n, r, and r^* are the number of vertices, notches, and resolved notches of P, respectively.*

PROOF. In each of the r^* iterations, the concavity of all notches is estimated and the notch with largest concavity is resolved; this takes $O(n)$ time. As no new notches are created, $r^* \leq r$ and the total time is $O(nr^*) = O(nr)$. □

The number of components in the final decomposition, m, depends on the tolerance τ and the smoothness of ∂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 17, respectively, generally have r close to $\Theta(n)$. When $r = \Theta(n)$, Chazelle and Dobkin's approach [7] has $O(n+r^3) = O(n^3)$ time complexity and Keil and Snoeyink's approach [18] has $O(n + r^2 \min\{r^2, n\}) = O(n^3)$ time complexity. In contrast, Algorithm 4.1 only requires $O(n^2)$ time to compute an exact convex decomposition.

7. EXPERIMENTAL RESULTS

7.1 Models and Implementation Details

We consider four polygonal models, maze, monkey, heron, and neuron; see Figures 15–18, respectively. The neuron has 18 holes, and the others have no holes.

We implemented our proposed algorithm in C++ and used FIST [14] as the triangulation subroutine for finding the shortest paths in pockets. Instead of resolving a notch r using a diagonal that bisected r's dihedral angle, we use a heuristic inspired by human perception. In particular, we score each possible diagonal for r using the following equation and pick the diagonal with the highest score.

$$f(r,x) = \begin{cases} 0 & : \overline{rx} \text{ does not resolve r} \\ \frac{(1+\alpha\times\text{concave}(x))}{(\beta\times\text{dist}(r,x))} & : \text{otherwise} \end{cases}$$

Here, $x \in \partial P_0$ and α and β are user defined scalars. According to experiments in [28], human vision prefers 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 α favors diagonals with large concavity and increasing β favors short diagonals. In our experiments, $\alpha = 0.1$ and $\beta = 1$ are used. This scoring process adds $O(n)$ time to each iteration and therefore does not change the overall asymptotic bound.

7.2 Experimentals

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 *decomposition time* for approximate convex decomposition (ACD) with different concavity measures and the minimum component exact convex decomposition (MCD) [18]. For a fair comparison, we re-coded the implementation available at [30] from Java to C++. Moreover, we estimate the *quality* of the final decomposition $\{C_i\}$ by measuring their *convexity*:

$$convex(\{C_i\}) = \frac{\sum_i \text{area}(C_i)}{\sum_i \text{area}(H_{C_i})}\ , \qquad (6)$$

where area(x) is the area of an object x and H_x is its convex hull. Eq. 6 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 computable significantly faster. For example, when $\tau = 0.1$ for the maze, monkey and heron models (see Figures 15-17), the ACD has 25%, 80%, or 50%, respectively, of the number of components in the MCD, and it was computed roughly 8-10 times faster. 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 17. 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 15) illustrates differences among the concavity measures. When $\tau > 10$, the convexity measurements in Figure 15(d) show that SL-concavity misses some important features that are found by SP-concavity (and thus also H-concavity). We also see that SP-concavity is more expensive to compute and that H-concavity is output sensitive.

The results for the larger monkey and heron models (Figures 16 and 17) 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 the bare eye. Moreover, the convexity of the 0.1-convex components of the monkey is 0.997 and of the heron is 0.98.

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

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"

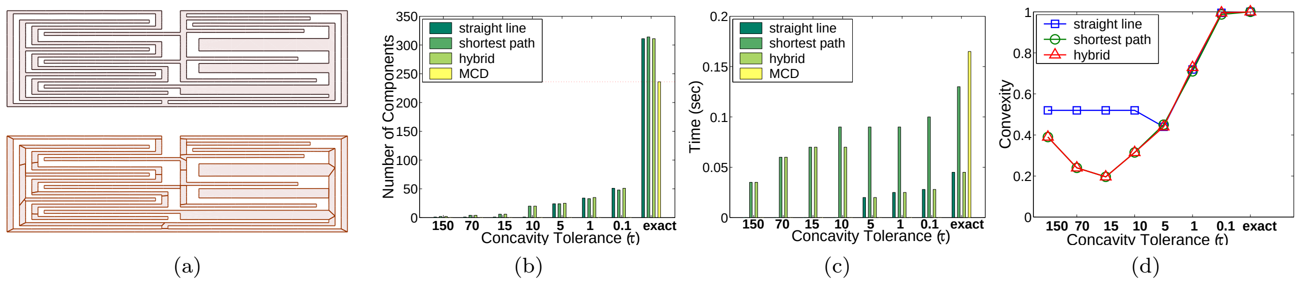


Figure 15. (a) The initial maze model (top) with 800 vertices and 400 notches, and an approximate convex decomposition (bottom). For several tolerance values, we show (b) the number of components in the final decomposition, (c) the decomposition time, and (d) convexity measurements.

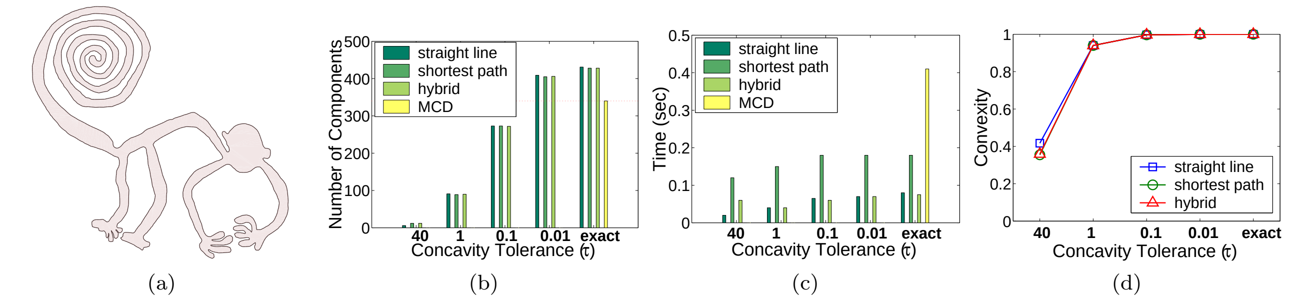


Figure 16. (a) The initial Nazca monkey has 1,204 vertices and 584 notches. For several tolerance values, we show (b) the number of components in the final decomposition, (c) the decomposition time, and (d) convexity measurements.

of the hole before the hole boundary is merged into the external boundary.

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 [21], and we are developing a method based on it for extracting 3D skeletons [20]. 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] 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.

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

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

[5] 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.

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

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

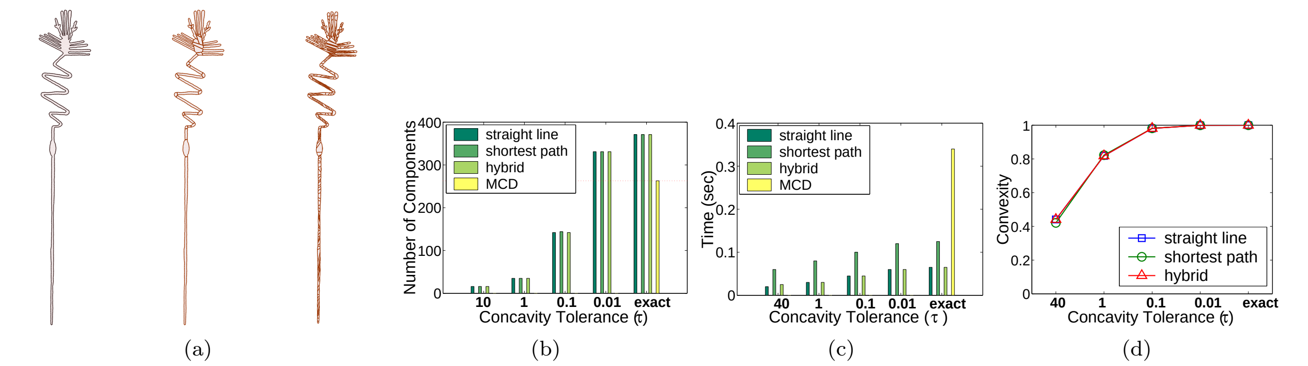


Figure 17. (a) The initial Nazca Heron model (left) has 1037 vertices and 484 notches. The radius of the bounding circle is 137.1. The decomposition using approximate convex decomposition (middle) has 49 components with concavity less than 0.5, and the optimal minimum convex decomposition (right) has 263 components. For several tolerance values, we show (b) the number of components in the final decomposition, (c) the decomposition time, and (d) convexity measurements.

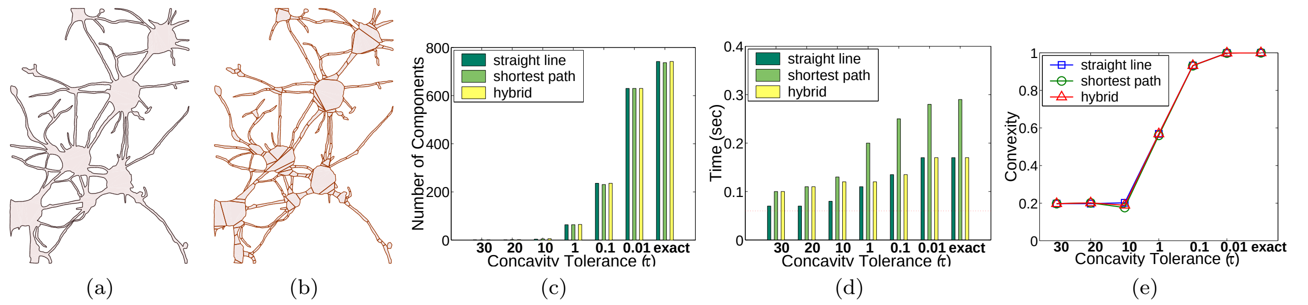


Figure 18. (a) The initial model of neurons has 1,815 vertices, 991 notches and 18 holes. The radius of the enclosing circle is 19.6. (b) The decomposition using approximate convex decomposition has 236 components with concavity less than 0.1. For several tolerance values we show (c) the number of components in the final decomposition, (d) the decomposition time (the dashed line indicates the time for resolving all holes), and (e) convexity measurements.

[17] 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, 1998.

[18] M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. *International Journal of Computational Geometry and Applications*, 12(3):181–192, 2002.

[19] 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.

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

[21] J.-M. Lien and N. M. Amato. Approximate convex decomposition. In *Proceedings of the 20th Annual ACM Symposium on Computational Geometry (SoCG 2004)*, 2004. Video Abstract. To appear.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

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

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

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

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

[32] 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.

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