In the Search for Good Neck Cuts

Anonymous author(s)
Anonymous affiliation(s)

Abstract

We study the problem of finding neck-like features on a surface. Applications for such cuts include robotics, mesh segmentation, and algorithmic applications. We provide a new definition for a surface bottleneck — informally, it is the shortest cycle relative to the size of the areas it separates. Inspired by the isoperimetric inequality, we formally define such optimal cuts, study their properties, and present several algorithms inspired by these ideas that work surprisingly well in practice. For examples of our algorithms, see `https://neckcut.space`.

2012 ACM Subject Classification Computing methodologies → Shape modeling

Keywords and phrases Constrictions, Object Representations, Computer Graphics

1 Introduction

Computing "good" cycles on surfaces is a well-studied problem [15, 7, 4, 6, 11, 16, 5, 3], such as computing a class of cycles, such as shortest geodesic cycles, non-contractable loops, handles, etc. We are interested in cycles that represent neck-like features on a surface. Identifying neck-like features on a 3D surface mesh has been a crucial algorithmic problem in applications such as robotics, mesh segmentation, and more. Neck-like cycles are often employed in intermediate steps within these applications, but computing them can be both challenging and time-consuming, as seen in [32, 25]. Existing methods tend to rely on expensive preprocessing for topological methods, which may also involve mesh modification, to produce neck-like cycles.

In this work, we consider two problems:

▶ **Problem 1.** *What is the optimal notion of a neck-like surface, and the cycles to define these necks?*

▶ **Problem 2.** *How can neck-like cycles be efficiently computed?*

We propose a new geometrically motivated definition of a *bottleneck curve* (or *neck-cut*), based on the isoperimetric quantity. We describe a theoretical approximation algorithm to find near-optimal bottleneck curves, which runs in polynomial time. We then implemented a practical algorithm, with this motivating background, to run on real models, which runs in sub-quadratic time with good results. Our practical algorithm is simple to implement, relying only on shortest path algorithms and filtering to achieve the results shown, with no second-pass optimization or curve smoothing required.

1.1 Background & Prior Work

Necks versus non-contractable loops. Finding neck-like features differs from finding the shortest non-contractable loops on a surface. As a reminder, a non-contractable loop on a surface corresponds to a cycle on the surface that can not be morphed into a point. Naturally, a neck-like loop might lie on an object that is topologically a ball (as are most of the examples shown in figures throughout this paper) – for example, on an hourglass, all the loops are contractable, yet it has a neck-like feature. While in many high-genus objects, these non-contractable loops may act as neck-like curves, there may be other non-contractable loops that do not lie on a feature boundary, or contractable loops that are on neck-like

features, which would not be considered. Finding non-contractible cycles can be used to reduce the genus of a surface, by cutting along them as shown in [15].

Sparsest cut (a.k.a. Cheeger's constant). A natural approach to addressing this problem is to consider the model as a graph $G = (V, E)$ and compute the sparsest cut. That is, find a partition of the vertices of G into two sets $S, \overline{S}$, such that the ratio

$$\phi(S, \overline{S}) = \frac{\left|E(S, \overline{S})\right|}{\min(|S|, |\overline{S}|)}$$

is minimized. Sadly, the associated optimization problem is NP-Hard, and instead one can use algebraic techniques to approximate it. People use various heuristics inspired by this observation to find good cuts. For example, Gotsman [18] noted the connection of the Cheeger constant to the Laplacian of a surface mesh. Using this, he was able to detect whether a small cut exists and how to partition the graph based on spectral embedding for genus-0 models.

However, the above definition of Cheeger's constant does not constrain that the cut is *connected*. Additionally, transitioning to algebraic methods often results in fuzzy boundaries that require further refinement. Gotsman's work approximates the optimal Laplacian basis, as computing the optimal would be on the order of $O(|V|^2)$ time.

These techniques appear to yield relatively slow algorithms. Since they do not work directly with the geometry, the generated cuts, while of relatively good quality, are not quite locally optimal.

Topological Methods. Abdelrahman and Tong [2] presented a method to compute neck-like features on meshes by locating critical points in a volumetric mesh and generating cutting planes over the mesh to isolate these loops. The primary observation was that essential points which were 2-saddles of a Morse function (generated by a distance function) would be good seed locations for neck-like features. This result was an extension of the method of Feng and Tong [16], which evaluated the persistent homology of the mesh to locate neck-like loops.

The method in [2] achieves a speedup from [16] by producing an initial neck loop from the cutting plane, which would eventually be smoothed out via shortest loop evaluation. The loops they generate are of good quality on all genus meshes. The main pitfall, however, is that they require a tetrahedral mesh to perform their algorithm, resulting in a substantial increase in vertex and face density. Other topological approaches, such as the one in [13], often have the same requirement of a volumetric representation of the mesh.

Surface Methods. Approaching this problem through topology is not the only route, however, as previous work has also considered the properties of the surface alone. Hétroy and Attali [23, 20, 21] compute geodesics on the surface, and slide to fit them to generate tight constrictions (neck-like features). Earlier works relied on mesh simplification to generate seed curves; however, in all cases, these algorithms rely on the local properties of geodesics to find neck loops.

Specifically, Hétroy [23] approximates the mean curvature of the mesh in all locations to find seed locations for constrictions. Then, the algorithm performs a local search from these seed locations until the constrictions are minimized, and smooths and minimizes the curve. The authors in [31] designed a fast algorithm to find shortest, exact geodesics on a model, regardless of the quality of the input mesh. However, initial cutting loops must be

specified in the input, as this algorithm was not meant for discovering loops from just the input model.

The authors in [30] use similar methods as our approach. However, when approximating constrictions, use the concavity of the curve. This algorithm requires computation of the Discrete Gaussian Curvature [26] of each point on a curve, requiring an $O(n^2)$ time algorithm to compute that metric.

1.2 Our approach

Our starting point is to define formally what constitutes a good neck cut. Intuitively, it is a curve that bounds a large area, while being short. In the plane, the largest area one can capture if the length of the perimeter is fixed is a disk. This innocent-looking observation is a consequence of the famous isoperimetric inequality. It states that for any region R in the plane, and any disk $\mathcal{d}$ of radius r, we have that

$$\frac{\mathsf{m}(R)}{\|\partial R\|^2} \leq \frac{\mathsf{m}(\mathcal{d})}{\|\partial \mathcal{d}\|^2} = \frac{\pi r^2}{(2\pi r)^2} = \frac{1}{4\pi},$$

where $\mathsf{m}(R)$ denotes the *area* of R, and $\|\eta\|$ denotes the *length* of a curve η. We view the ratio on the left as the *isoperimetric ratio* of the boundary of R. A good neck-cut would have a high ratio. A curve on a surface might bound a much larger area compared to its perimeter, but unlike the plane, we have to consider both sides bounded by the curve. As such, the *tightness* of a closed curve on a surface (of genus zero) is the minimum, among the two regions it bounds, of the isoperimetric ratio.

Computing tightness. Unfortunately, computing (or even approximating) the tightest cycle on a surface appears to be hopeless in terms of efficient algorithms. Nevertheless, it provides us with an easily computable scoring function to compare cycles (i.e., the tighter, the better). There are cases where the optimal neck-cut is intuitively obvious, see Figure 2. We thus investigate sufficient conditions under which we can efficiently approximate the optimal neck-cut. To this end, we first formally define tightness in Section 2.1.

Specifically, for a loop, we look at the ratio between the area it encloses and its length squared (for a circle in the plane, this ratio is a constant). Clearly, the bigger the ratio, the better neck-like the cycle is. We formally define the underlying optimization problem in Section 2.1.

Well-behaved surfaces, salient points, and discovering necks. We quantify, in Section 2.2, what it means for a surface to be well-behaved – intuitively, it should have bounded growth (which all real-world surfaces seem to possess). To discover the neck-cuts, we try to identify necks – to this end, we study in Section 2.3 *salient* points that can be used to define necks. Intuitively, salient points are extremal points of the model (such as the tips of fingers in a human model). The paths connecting distant salient points (such as the path between the tip of a finger and the tip of a toe of a human model) can then be used to identify (implicitly) necks that should contain good cuts.

Approximation algorithms In Section 3, we present an efficient approximation algorithm to the optimal collar (i.e., best neck-cut) under certain (pretty strong) conditions. This approximation algorithm gives us reasonable bounds for the total time complexity required, while also motivating the core heuristics used in the practical algorithm.

In Section 4, we discuss a surface-based practical algorithm. Our algorithm uses the salient points as a baseline on which a neck-like cycle must lie on the path between salient points. Because of this, we can mainly rely on shortest path algorithms with little preprocessing and focus on filtering cycles of interest.

Our practical algorithm is based on a few heuristics derived from the properties of bottleneck curves, and thus can produce good bottleneck curves in surface meshes. We discuss the performance of our algorithm and the viability of generating these curves in a real-time setting. Unlike previous work, our algorithm avoids complex global computation or iterative smoothing.

Numerous examples of the output of our algorithm are provided at `https://neckcut.space` and discussed in Section 5.

2 Isoperimetry, bottleneck cuts, and salient points

2.1 Isoperimetry and bottlenecks

Isoperimetric problem on surfaces. The isoperimetric problem asks to determine a plane figure of maximum area, with a specified boundary length. This problem dates back to antiquity, but a formal solution was not provided until the 19th century. It is known [8] that circles, and in higher dimensions balls, are the optimal shapes. Even in the plane, proving it was quite a challenge. For a planar closed curve σ, consider its *isoperimetric ratio*:

$$\rho(\sigma) = \frac{\mathsf{m}(\mathrm{int}(\sigma))}{\|\sigma\|^2},$$

where $\mathrm{int}(\sigma)$ is in the interior region bounded by σ, and $\|\sigma\|$ is the length of σ. This ratio can be arbitrarily small (i.e., consider a wiggly shape that has a small area but a long boundary). The isoperimetric inequality states that this ratio is maximized for the disk, where it holds with equality. Namely, the isoperimetric inequality states that, for any closed planar curve σ, we have $\rho(\sigma) \leq \frac{1}{4\pi}$.

On a finite surface (say of genus zero) in 3D, it is natural to try to compute a closed curve on the surface as short as possible that splits the surface area into two "large" parts. As a concrete example, consider the natural cycle in the base of a human finger – it does not partition the surface (i.e., a human model) even remotely equally. And yet, it is intuitively a good neck-cut.

Tightness. To overcome this for a surface $\mathcal{M}$ (say of genus 0), we define a variant of the isoperimetric ratio.

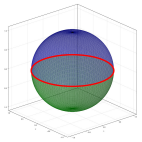
▶ **Definition 3.** *For a surface $\mathcal{M}$ in $\mathbb{R}^3$ of genus zero, and a region $\mathfrak{b} \subseteq \mathcal{M}$, let the* **tightness** *of $\mathfrak{b}$ is the ratio*

$$\langle \mathfrak{b} \rangle = \frac{\min\big(\mathsf{m}(\mathfrak{b}), \mathsf{m}(\overline{\mathfrak{b}})\big)}{\|\partial b\|^2},$$

where $\partial\mathfrak{b}$ is the boundary *of $\mathfrak{b}$, $\mathsf{m}(\mathfrak{b})$ is the* area *of $\mathfrak{b}$, the complement of $\mathfrak{b}$ is $\overline{\mathfrak{b}} = \mathcal{M} \setminus \mathfrak{b}$, and $\|\partial\mathfrak{b}\|$ denotes the length of $\partial\mathfrak{b}$. In particular, for a close curve η, that splits the surface into two parts $\mathfrak{b}$ and $\overline{\mathfrak{b}}$, its* **tightness** *$\langle\eta\rangle$ is the tightness of $\psi(\mathfrak{b})$.*

Here, the closed curve is $\partial\mathfrak{b}$, the patches generated by $\partial\mathfrak{b}$ are $\mathfrak{b}$ and $\overline{\mathfrak{b}}$, and its tightness is "roughly" the isoperimetric ratio. It is thus natural to ask for the patch $b \subseteq \mathcal{M}$ with maximum tightness.

▶ **Example 4.** Consider the $\mathcal{M}$ to be a unit radius sphere in 3D. It is not hard to see that the maximum tightness is realizable by $\mathcal{b}$ being the (say, top) hemisphere of $\mathcal{M}$, and $\partial\mathcal{b}$ being the equator. In that case, $\mathsf{m}(\mathcal{b}) = 2\pi$, and $\psi(\mathcal{b}) = 2\pi/(2\pi)^2 = \frac{1}{2\pi}$. Intuitively, closed curves are "interesting" as far as being a neck-cut if their tightness is at least $\frac{1}{2\pi}$. (Compare this to the disk in the plane, that has tightness $\frac{1}{4\pi}$.)



In general, proving that a specific patch $b \subseteq \mathcal{M}$ is the one realizing maximum tightness is a challenging problem, as the long history of the isoperimetric inequality testifies [27, 10, 22]. A bottleneck curve on a surface should have the property that it is short, while enclosing a large area on both sides (e.g., a neck of an hourglass). Thus, our proxy for finding a good bottleneck curve is going to be computing curves on a given surface that have high tightness.

▶ **Problem 5.** *Given a surface $\mathcal{M}$, compute a region b with tightness $\psi(b)$ as large as possible.*

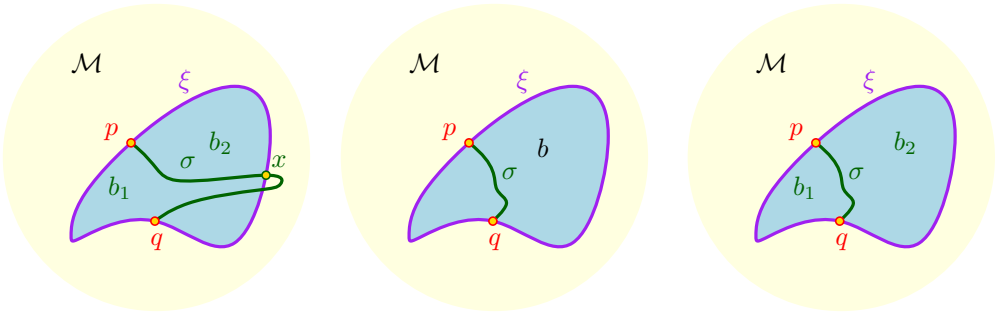


Figure 1 Middle, Right: For some optimal bottleneck ξ, we consider the geodesic σ. Left: If the geodesic were to cross ξ at x, it would be a contradiction, as shortcutting along ξ would be shorter.

▶ **Definition 6.** *Let $\mathcal{M}$ be a surface of genus zero, and let ξ be a cycle on $\mathcal{M}$. Let $\mathcal{b}$ be the region bounded by ξ on $\mathcal{M}$. The cycle ξ is a* **α-bottleneck** *of $\mathcal{M}$, if the tightness of $\mathcal{b}$ is at least α. The α-bottleneck with maximum α on $\mathcal{M}$, is the* optimal bottleneck cut*, or simply a* **collar**.

2.2 Well behaved surfaces

To explain some intuition for the heuristics used in Section 4, we discuss a few properties of a *well-behaved* surface.

Slow-expansion on the surface. We are interested in surfaces such that their measure (i.e., area/volume) does not expand too quickly.

To this end, given a set σ on a $\mathcal{M}$, its *r-expansion* is the region

$$\sigma \oplus r = \{p \in \mathcal{M} \mid \mathsf{d}_{\mathcal{M}}(p, \sigma) \leq r\}. \tag{1}$$

▶ **Definition 7.** *A model $\mathcal{M}$ is* **τ-expanding**, *if for any curve $\sigma \subseteq \mathcal{M}$, and any $r \geq 0$, we have that $\mathsf{m}(\sigma \oplus r) \leq \tau(\|\sigma\| r + r^2)$ and $\|\partial(\sigma \oplus r)\| \leq \tau(\|\sigma\| + r)$. The minimum such τ is the* expansion *of $\mathcal{M}$, denoted by $\tau^\star$.*

▶ **Example 8.** In the plane, Steiner inequality [19] implies that for any curve σ we have $\mathsf{m}(\sigma \oplus r) \leq \pi r^2 + 2r\,\|\sigma\|$ and $\|\partial(\sigma \oplus r)\| \leq 2\,\|\sigma\| + 2\pi r$. Thus, the plane is 2π-expanding.

▶ **Definition 9.** *A simple cycle σ is* **contractible** *if one can continuously morph σ to a single point. A portion $\mathcal{R} \subseteq \mathcal{M}$, and a cycle $\sigma \subset \mathcal{R}$, the cycle is* **$\mathcal{R}$-contractible**, *if it can be contracted to a point, while staying inside $\mathcal{R}$.*

Note that while any cycle σ on the original surface $\mathcal{M}$ is contractible, if $\mathcal{M}$ has genus zero, it might not be $\mathcal{R}$-contractible – for example, if $\mathcal{R}$ is the result of taking $\mathcal{M}$ and creating two punctures on both sides of σ. Intuitively, tight cycles are not locally contractible – one has to go far to be able to collapse them to a point.

▶ **Definition 10.** *For $\mathcal{R} \subseteq \mathcal{M}$, and an $\mathcal{R}$-contractible cycle $\sigma \subseteq \mathcal{R}$, let $\mathcal{b}_\sigma \subseteq \mathcal{R}$ be the portion of $\mathcal{M}$ bounded by a closed curve σ (the other portion of $\mathcal{M}$ bounded by σ might contain portions outside $\mathcal{R}$). If $\mathcal{R} = \mathcal{M}$, let $\mathcal{b}_\sigma$ denote the smaller area patch (out of the two patches) induced by σ on $\mathcal{M}$. The region $\mathcal{R}$ is* **α-well-behaved** *if for all $\mathcal{R}$-contractible cycles $\sigma \subset \mathcal{R}$, we have that $\mathsf{m}(\mathcal{b}_\sigma) \leq \alpha \|\sigma\|^2$.*

▶ Remark 11. Consider a region $\mathcal{R} \subseteq \mathcal{M}$, where $\mathcal{M}$ is τ-expanding, such that any $\mathcal{R}$-contractible cycle σ in it, is $\big((\sigma \oplus r) \cap \mathcal{R}\big)$-contractible, where $r = \tau\|\sigma\|$. Then, the τ-expansion implies that $\mathsf{m}(\mathcal{b}_\sigma) \leq \mathsf{m}(\sigma \oplus r) \leq \tau(\|\sigma\| r + r^2) \leq 2\tau^3 \|\sigma\|^2$). Namely, $\mathcal{R}$ is $2\tau^3$-well behaved.

2.3 Salient points to a bottleneck

In the following, we assume the given surface $\mathcal{M}$ is triangulated, has genus 0, and it has a useful collar (i.e., α-tight for a "large" α). In addition, we assume $\mathcal{M}$ is τ-expanding, where τ is some small constant. Let $\mathcal{o}$ denote this optimal α-bottleneck of $\mathcal{M}$. The cycle $\mathcal{o}$ breaks $\mathcal{M}$ into two regions $\mathcal{b}$ and $\overline{\mathcal{b}}$. Let $\mathsf{s}(\mathcal{o}, \mathcal{b})$ be the point furthest away from $\mathcal{o}$ on b. Formally, we define

$$\mathsf{s}(\mathcal{o}, \mathcal{b}) = \arg\max_{p \in b} \mathsf{d}_{\mathcal{M}}(p, \mathcal{o}) \qquad \text{where} \qquad \mathsf{d}_{\mathcal{M}}(p, \mathcal{o}) = \min_{q \in \mathcal{o}} \mathsf{d}_{\mathcal{M}}(p, q).$$

Such points are ***salient***, and they are far from the bottleneck if the surface is well behaved.

▶ **Lemma 12** (salient points are far). *For $s = \mathsf{s}(\mathcal{o}, \mathcal{b})$, we have that $\mathsf{d}_{\mathcal{M}}(s, \mathcal{o}) \geq \|\mathcal{o}\|$, if $\alpha \geq 4\tau$.*

Proof. Let $\ell = \|\mathcal{o}\|$. By $\mathcal{o}$ being an α-bottleneck, we have that

$$\mathsf{m}(\mathcal{b}) \geq \alpha \|\mathcal{o}\|^2 .$$

On the other hand, for $r = \mathsf{d}_{\mathcal{M}}(s, \mathcal{o})$, we have $b \subseteq \mathcal{o} \oplus r$. By the τ-expansion of $\mathcal{M}$, we have

$$\mathsf{m}(\mathcal{b}) \leq \mathsf{m}(\mathcal{o} \oplus r) \leq \tau(\|\mathcal{o}\| r + r^2).$$

Thus, we have $\alpha \|\mathcal{o}\|^2 \leq \tau(\|\mathcal{o}\| r + r^2) \leq \tau(\|\mathcal{o}\|/2 + r)^2$. This implies that

$$\frac{\alpha}{\tau} \|\mathcal{o}\|^2 \leq (\|\mathcal{o}\|/2 + r)^2 \qquad \Longrightarrow \qquad r \geq \left(\sqrt{\frac{\alpha}{\tau}} - \frac{1}{2}\right) \|\mathcal{o}\|$$

as $\alpha/\tau \geq 4$. ◀

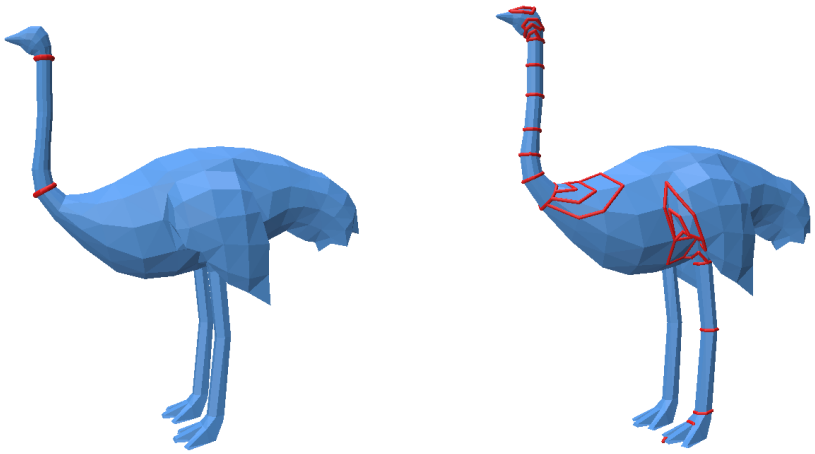


Figure 2 Left: A long neck with a stable collar. Right: Every geodesic cycle from the beak to the left foot.

3 Approximating the optimal collar

3.1 Identifying the neck where the collar lies

Consider the easy case, that not only is there a good collar, but this collar is stable, in the sense that one can slide it up and down the "neck" and the quality of the collar remains relatively the same, see Figure 2.

▶ **Definition 13.** *Let κ be a "long" shortest path on $\mathcal{M}$ with endpoints s and t. For every point $p \in \kappa$, consider the shortest path $\circlearrowleft_\kappa(p)$ from p to itself, if we were to cut the surface $\mathcal{M}$ along κ, and the path σ has to connect p to its copy. The closed curve $\circlearrowleft_\kappa(p)$ is a* **lasso** *if*

$$\|\circlearrowleft_\kappa(p)\| > \max\bigl(\mathsf{d}_{\mathcal{M}}(p,s), \mathsf{d}_{\mathcal{M}}(p,t)\bigr),$$

and is denoted by $\circlearrowleft_\kappa(p)$.

▶ **Example 14.** Let $\mathcal{M}$ be the surface of the following solid – connect two large disjoint balls by a thin and long cylinder (i.e., a dumbbell) – see Figure 3. Consider the cylinder portion of the surface – it forms a natural neck, and let $\mathcal{R}$ denote it. Any curve going around the neck is not contractible on the neck, while any closed curve σ that is contractible on the neck, is going to have area $O(\|\sigma\|^2)$. That is, the neck is $O(1)$-well behaved.

▶ **Observation 15.** *Two lassos defined using the same base path π can not cross each other.*

▶ **Definition 16.** *Consider two lassos τ_1, τ_2 defined using a base path (which is a shortest path) π. The* **neck** *$\mathcal{N} = \mathcal{N}(\tau_1, \tau_2)$ is the area on the surface $\mathcal{M}$ lying between τ_1 and τ_2. Such a region is* **β-neck** *if it is β-well-behaved, for some $\beta > 0$.*

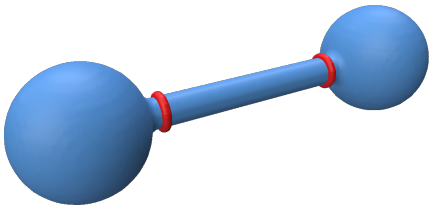


Figure 3

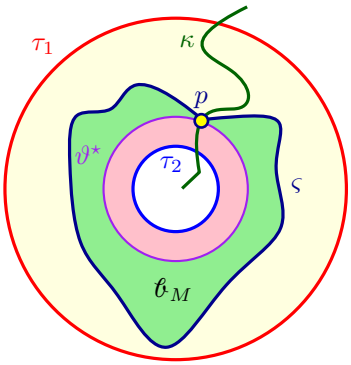


Figure 4

▶ **Lemma 17.** *Let s, t be two points on $\mathcal{M}$, and let κ be the shortest path connecting them. Let τ_1, τ_2 be two lassos defined by points on κ, and let $\mathcal{N} = \mathcal{N}(\tau_1, \tau_2)$ be the induced α-neck. Assume that the optimal collar $\vartheta^\star$ is contained in $\mathcal{N}$, and its tightness $\beta \geq 8\alpha$. Then, there exists a point $p \in \kappa \cap \mathcal{N}$, such that its lasso $\varsigma = \circlearrowleft_\kappa(p)$ is $\beta(1 - \frac{4\alpha}{\beta})$-tight.*

Proof. The algorithm picks a point $p \in \kappa \cap \vartheta^\star$ as the base point for the construction. Let $\varsigma = \circlearrowleft_\kappa(p)$ be the associated lasso.

Assume, for now, that the lasso ς only intersects $\vartheta^\star$ at p (as in Figure 4). In that case, ς and $\vartheta^\star$ are homotopic, and let $\mathscr{b}_i$ be the area in between them. By the α-behaveness of $\mathcal{N}$, and since $\partial\mathscr{b}_i$ is $\vartheta^\star \cup \varsigma$ (and is $\mathcal{N}$-contractible), we have that

$$\mathsf{m}(\mathscr{b}_i) \leq \alpha(\|\vartheta^\star\| + \|\varsigma\|)^2 \leq 4\alpha \|\vartheta^\star\|^2 ,$$

as $\|\varsigma\| \leq \|\vartheta^\star\|$ – indeed, $\vartheta^\star$ is a candidate for the shortest path connecting p to itself "around" κ, but ς is the shortest one.

For a closed curve η, let $\mathscr{b}_\eta$ and $\overline{\mathscr{b}}_\eta$ be the two parts of $\mathcal{M}$ bounded by η. Observe that

$$B = \min\big(\mathsf{m}(\mathscr{b}_\varsigma), \mathsf{m}(\overline{\mathscr{b}}_\varsigma)\big) \geq \min\big(\mathsf{m}(\mathscr{b}_{\vartheta^\star}), \mathsf{m}(\overline{\mathscr{b}}_{\vartheta^\star})\big) - \mathsf{m}(\mathscr{b}_i).$$

The tightness of ς is thus $\langle\varsigma\rangle = \dfrac{B}{\|\varsigma\|^2} \geq \dfrac{\min\big(\mathsf{m}(\mathscr{b}), \mathsf{m}(\overline{\mathscr{b}})\big) - \mathsf{m}(\mathscr{b}_i)}{\|\vartheta^\star\|^2} \geq \langle\vartheta^\star\rangle - 4\alpha.$

The slightly harder case is when ς and $\vartheta^\star$ have several intersections. In that case, the $\varsigma \cup \vartheta^\star$ forms an arrangement – the "inner" region/face denoted by $\mathscr{b}_s$ and the outer face denoted by $\mathscr{b}_t$. So consider all the other faces $f_1, \ldots, f_k$ in this arrangement. These faces are all contractible disks. Let ℓ_i be the boundary of the ith face, for $i = 1, \ldots, k$. Observe that every edge e of ς or $\vartheta^\star$ contributes at most $2\|e\|$ to the total lengths of these boundary faces. Thus, we have $\sum_{i=1}^{k} \ell_i \leq 2(\|\varsigma\| + \|\vartheta^\star\|) \leq 2\|\vartheta^\star\|$. Setting $\mathscr{b}_i = (\mathscr{b}_{\vartheta^\star} \setminus \mathscr{b}_\varsigma) \cup (\mathscr{b}_\varsigma \setminus \mathscr{b}_{\vartheta^\star})$ to be

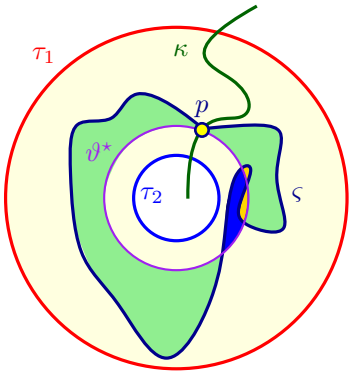


Figure 5

region that is the symmetric difference between $\mathcal{b}_{\vartheta^\star}$ and $\mathcal{b}_\varsigma$, we have

$$\mathsf{m}(\mathcal{b}_i) \leq \sum_{t=1}^{k} \mathsf{m}(f_t) \leq \sum_{t=1}^{k} \alpha \ell_t^2 = \alpha \sum_{t=1}^{k} \ell_t^2 \leq \alpha \Big(\sum_{t=1}^{k} \ell_t \Big)^2 4\alpha \left\| \vartheta^\star \right\|^2 .$$

The claim now follows from the argument above, and observing that $\langle \varsigma \rangle \geq \langle \vartheta^\star \rangle - 4\alpha = \beta\big(1 - \frac{4\alpha}{\beta}\big)$. ◀

▶ **Corollary 18.** *In the settings of Lemma 17, if the tightness β of the optimal collar on an α-neck $\mathcal{N}$ is $\geq 4\alpha/\varepsilon$, for $\varepsilon \in (0,1)$, then there is lasso on $\mathcal{N}$ of tightness $\geq (1-\varepsilon)\beta$. Namely, the lasso has tightness ε-close to optimal.*

3.2 The algorithm for computing the collar

Lemma 17 implies that if the optimal tightness is much bigger than the α (the well-behavedness of the neck), then one can efficiently compute a collar with tightness close to optimal. Importantly, such a lasso is efficiently computable.

We outline here the basic idea – our purpose is to present a polynomial-time approximation algorithm. To this end, we guess the points s and t of Lemma 17, and compute the shortest path κ between them. Next, we guess the two points, $x, x' \in \kappa$, defining the lassos bounding the optimal collar $\vartheta^\star$, and we compute these two lassos. We compute the region $\mathcal{N}$ bounded in between τ_1 and τ_2, and verify that it indeed has the topology of a sleeve (this can be done by example by computing its Euler characteristic, and verifying that τ_1 and τ_2 cover all the boundary edges of this patch. Now, one can try to compute the shortest path around the neck for each vertex $v \in \kappa \cap \mathcal{N}$, and explicitly determine its tightness. The maximum one found is the desired approximation. If the model has size n, the running time of this algorithm is $O(n^5)$. We thus get the following.

▶ **Theorem 19.** *Let $\mathcal{M}$ be a triangulated surface in 3D with genus 0 and n vertices. Assume the optimal collar $\vartheta^\star$ on $\mathcal{M}$ lies on an α-neck $\mathcal{N}$ that is induced by a shortest path κ, and two lassos τ_1, τ_2 (see Lemma 17). Furthermore, the tightness $\langle \vartheta^\star \rangle \geq 8\alpha$. Then, one can compute, in $O(n^5)$ time, a closed curve ς such that $\langle \varsigma \rangle \geq \langle \vartheta^\star \rangle - 4\alpha \geq \langle \vartheta^\star \rangle / 2$.*

▶ Remark 20. The above algorithm inspires our practical algorithm, which achieves sub-quadratic running time by avoiding the need to guess all pertinent information. Thus, we had not spent energy on improving the running time of Theorem 19. In particular, the stated running time should be taken as evidence that the optimal collar can be approximated efficiently under certain conditions.

4 A Practical Algorithm

We will discuss an implementation that utilizes a few heuristics to locate optimal bottleneck curves while maintaining fast performance. This algorithm is an approximation and may not find all optimal curves. However, as we have implemented it, it is the fastest solution, and optimizations are possible to produce high-quality results in real-time. We assume and run on models of genus 0.

The process can be described in two stages for any given mesh: In the first stage, we locate salient points that lie on the tips of features. In the second stage, we take paths between salient points and search for bottleneck curves. Here, we discuss the details of each process and what steps we take to return relevant curves. In Section 5, we discuss how this algorithm performs on various meshes in the wild.

4.1 Finding Salient Points

We aim to identify distant pairs of *salient points* to search for bottleneck curves. Usually, these points represent the tips of various features on a mesh (such as the tip of a spike, finger, or head) but do not necessarily lie on the convex hull of the mesh. Finding exact salient points can be difficult and time-consuming. We will use a standard method to locate points that are near "true" salient points, but are sensitive to the starting location of our search. Previous work has focused on identifying salient points and utilizing them in mesh decomposition [33]. Other methods have also utilized feature points and surface methods to perform mesh decomposition [14, 17, 24, 12]. We use shortest path algorithms to locate these salient points, which act as reasonable estimates for the exact salient points that would be used to find bottleneck cuts.

Let T_s be the shortest-path tree rooted at s, let $d(v)$ be the distance from s to v. We first compute a point on the mesh that lies on a 2-approximation of the diameter of the mesh:

1. Pick an arbitrary point on the graph s.
2. Compute the shortest-path tree T_s. Let u be the leaf with the distance from s.
3. Compute shortest-path tree T_u. Let v be the leaf with maximum distance from u.

We consider the shortest path tree T_u. A vertex x is a ***salient point*** if it is a local maximum of its neighbors with respect to distance to u. That is, for every neighbor y, $\mathsf{d}_{\mathcal{M}}(u, y) < \mathsf{d}_{\mathcal{M}}(u, x)$. Because v is the furthest point from u, all its neighbors must have a shorter distance, and thus it is a salient point. For every leaf on T_u, we check all its neighbors on the mesh and mark it as a salient point if the above condition holds. Let C denote the set of salient points.

Depending on the quality of the mesh, some filtration of salient points may be necessary. Extremely noisy surfaces can create several local maximum points close together. In these cases, we want to eliminate any salient point within a user-selected distance r from any salient point. We sort C and perform an r-depth breadth-first search from each salient point, removing any salient points from C that we encounter. In practice, r is no larger than a small constant to handle small perturbations in a mesh. See Figure 6 for an example.

4.2 Finding Bottleneck Curves

Once we have found a set of salient points C, we can begin the process of locating bottleneck curves. We will build a simple skeleton of the mesh. We start with u and v from the previous

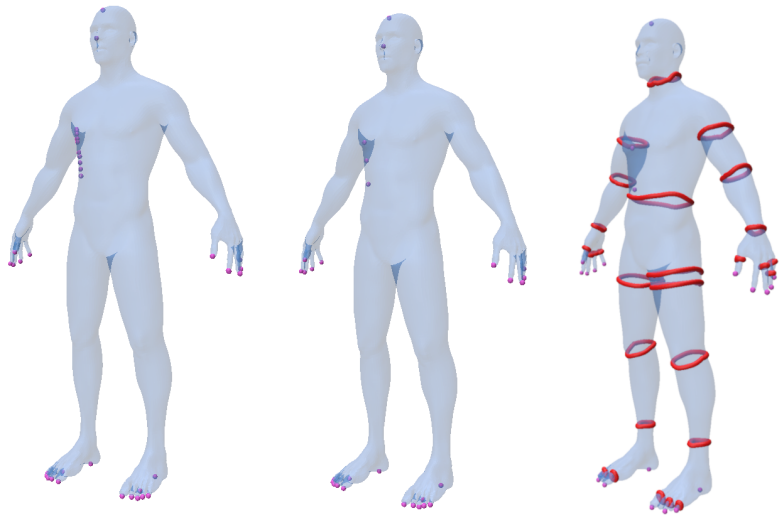


Figure 6 The salient points of a human mesh, with no filtering (left), $r = 10$ (middle), and $r = 20$ (right). The bottleneck curves are shown in the right mesh.

section and the path between them. For each candidate x, we find a path between x and the path between u and v. We add this path to our skeleton. Each remaining candidate will find the shortest path between it and the constructed skeleton, and we repeat this process until all candidates are connected. Let this skeleton be K.

For each path $\pi \in K$, we can find a cycle on this skeleton by running a shortest-path algorithm from some vertex $v \in \pi$ to itself, where no vertices on π can be used, thus preventing the shortest-path algorithm from crossing π. This cycle is a geodesic (with respect to π) cycle which includes v. This process generates an ordered sequence of cycles Π. We want to select cycles with a local maximum tightness. To compute the tightness requires computing the area bounded by each cycle $\xi \in \Pi$, and selecting local maxima within Π. On genus zero objects, we can compute the area between every adjacent pair of cycles (e.g. $\xi_i, \xi_{i+1} \in \Pi$ for each i), and then use prefix sums to compute the area bounded by one cycle efficiently. For a cycle ξ, if its tightness is the local maximum among its neighbors, then it is likely a good bottleneck.

Cycles near boundaries and salient points might be of low quality. Likewise, cycles that bound a small amount of area, whilst also being short, can lead to other cycles which appear to be local maxima among their neighbors. To that end, we refer to the calculation in Example 4, and filter out any cycle with a computed tightness less than $\frac{1}{2\pi}$ (with some tolerance due to inaccuracies with triangulated meshes). These cycles aren't representative of a large enough bounded area, and thus do not make good bottleneck curves.

4.3 Timing Analysis

Computing the set of candidates takes $O(n \log n)$ time, for running shortest paths, and the r-depth filtering, which is no worse than $O(n)$ time. Let $k = |C|$ and $|K|$ be the number of vertices in the skeleton. Computing the skeleton itself takes $O(cn \log n)$ time. The bulk of our time results from computing the cycles along the paths, which takes $O(|K| n \log n)$ time. In practice, this is much faster, since we are running an st-shortest path algorithm, which

terminates early.

There are additional optimizations in practice that can be employed that would further speed up the cycle searching. Mainly, computing the cycles along each path is not reliant on data from the other paths in K. Namely, with parallelism, each path can be computed independently, thus bounding the runtime based on the number of available cores and the longest path within the skeleton.

5 Evaluation

We implemented our algorithm in C++, using the Polyscope & Geometry Central [29, 28] libraries. These libraries provide a halfedge data structure, with standard traversal algorithms. All timings were measured on a single thread of an Intel i7-14700K 3.4GHz CPU. We tested our algorithm on *Benchmark for 3D Mesh Segmentation Dataset* [9], along with additional meshes. All of our results from this dataset, along with the code, can be viewed on meshcuts.space. We have selected a few to feature in this paper, as shown in Table 1. We measured the runtime of all genus zero models, as shown in Figure 7. We attempted to use the code from [2] for comparison. However, we were unable to reproduce a working program, despite repeated efforts.

Input	Description	Faces
7	Public Domain: Human 1	48918
2	MSB/Stanford: Armadillo	50542
3	MSB: Octopus	28248
4	MSB: Ant	13696
5	MSB: Horse	11072
6	MSB: Hand	3026
1	MSB: Human 2	11258
8	Stanford: Bunny	69630

Table 1 Selected inputs from our testing. MSB models are from [9], Stanford models are from [1]

This implementation still has some optimizations to be implemented, as described in Section 4.3. The timings described in Table 2 are single-threaded operations. We lazily compute the bounded area in this implementation, only running the prefix-sum method per path, rather than the entire skeleton at once. However, further optimizations would not have a significant impact on the runtime. The results from these models can be seen in Figure 8, Figure 9, and Figure 10.

When deciding which cycles to display, we chose cycles whose tightness is a local maximum among a window of five cycles. From the discussion in Section 3, a neck is defined by two collars, with some optimal collar lying within the neck. In practice, we observe that several cycles all reach a maximum tightness as a group, since these neck-like surfaces are usually well-behaved. In dense meshes, such a small exclusion window would result in several cycles with similar tightness reported together. Alternate implementations can choose to report all or some of these cycles, but in either case, the same neck-like feature is identified.

6 Conclusions

We proposed a new definition for neck-like features and the curves that defined them. We also presented an approximation and practical algorithm to detect these bottleneck curves on

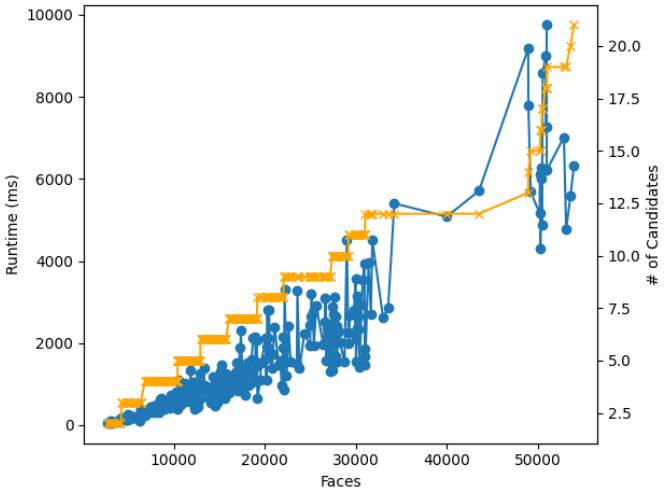


Figure 7 Runtime of models from [9]. Runtime is plotted with *blue* dots. The number of candidates is plotted with *orange* crosses.

real-world meshes. We believe our method has improvements over previous work [30, 23, 2], while also being extremely simple to implement. For future work, we wish to explore using this algorithm in other applications. One possible application is to use this algorithm as the seeds for [31], rather than user-defined cutting planes, to discover exact geodesics from our output cycles.

References

1 The Stanford 3D Scanning Repository — graphics.stanford.edu. https://graphics.stanford.edu/data/3Dscanrep/.

2 Hayam Abdelrahman and Yiying Tong. Fast Computation of Neck-Like Features. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5384–5393, December 2023. Conference Name: IEEE Transactions on Visualization and Computer Graphics. URL: https://ieeexplore.ieee.org/document/9910018, doi:10.1109/TVCG.2022.3211781.

3 Florian Beguet, Sandrine Lanquetin, and Romain Raffin. Flexible mesh segmentation via Reeb graph representation of geometrical and topological features. *CoRR*, abs/2412.05335, 2025. arXiv:2412.05335, doi:10.48550/arXiv.2412.05335.

4 Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Reeb graphs for shape analysis and applications. *Theor. Comput. Sci.*, 392(1-3):5–22, 2008. URL: https://doi.org/10.1016/j.tcs.2007.10.018, doi:10.1016/J.TCS.2007.10.018.

5 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. *J. Comput. Geom.*, 7(1):123–148, 2016. URL: https://doi.org/10.20382/jocg.v7i1a7, doi:10.20382/JOCG.V7I1A7.

6 Sergio Cabello, Matt DeVos, Jeff Erickson, and Bojan Mohar. Finding one tight cycle. *ACM Trans. Algorithms*, 6(4):61:1–61:13, 2010. doi:10.1145/1824777.1824781.

7 Sergio Cabello and Bojan Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discret. Comput. Geom.*, 37(2):213–235, 2007. URL: https://doi.org/10.1007/s00454-006-1292-5, doi:10.1007/S00454-006-1292-5.

8 Lamberto Cesari. Rectifiable Curves and the Weierstrass Integral. *The American Mathematical Monthly*, 65(7):485–500, 1958. Publisher: [Taylor & Francis, Ltd., Mathematical Association of America]. URL: https://www.jstor.org/stable/2308574, doi:10.2307/2308574.

Figure 8 Inputs 1 & 2

9 Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

10 Andrew Cotton, David Freeman, Andrei Gnepp, Ting Ng, John Spivack, and Cara Yoder. The isoperimetric problem on some singular surfaces. *Journal of the Australian Mathematical Society*, 78(2):167–197, April 2005. URL: https://www.cambridge.org/core/product/identifier/S1446788700008016/type/journal_article, doi:10.1017/S1446788700008016.

11 Tamal K. Dey, Fengtao Fan, and Yusu Wang. An efficient computation of handle and tunnel loops via reeb graphs. *ACM Trans. Graph.*, 32(4):32:1–32:10, 2013. doi:10.1145/2461912.2462017.

12 Tamal K. Dey, Joachim Giesen, and Samrat Goswami. Shape segmentation and matching with flow discretization. In Frank Dehne, Jörg-Rüdiger Sack, and Michiel Smid, editors, *Algorithms and Data Structures*, pages 25–36, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

13 Tamal K. Dey, Kuiyu Li, Jian Sun, and David Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3D models. *ACM Transactions on Graphics*, 27(3):1–9, August 2008. URL: https://dl.acm.org/doi/10.1145/1360612.1360644, doi:10.1145/1360612.1360644.

14 Vladislav Dordiuk, Maksim Dzhigil, and Konstantin Ushenin. Surface Mesh Segmentation Based on Geometry Features. In *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 270–273, May 2023. ISSN: 2769-3635. URL: https://ieeexplore.ieee.org/document/10158888, doi:10.1109/USBEREIT58508.2023.10158888.

15 Jeff Erickson and Sariel Har-Peled. Optimally Cutting a Surface into a Disk. *Discrete Comput. Geom.*, 31(1):37–59, January 2004. doi:10.1007/s00454-003-2948-z.

16 Xin Feng and Yiying Tong. Choking Loops on Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1298–1306, August 2013. Conference Name: IEEE Transactions on Visualization and Computer Graphics. URL: https://ieeexplore.ieee.org/document/6409845, doi:10.1109/TVCG.2013.9.

17 Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3D models. *Computers & Graphics*, 33(3):262–269, June 2009. URL: https://www.sciencedirect.com/science/article/pii/S0097849309000454, doi:10.1016/j.cag.2009.03.010.

18 C. Gotsman. On graph partitioning, spectral analysis, and digital mesh processing. In *2003 Shape Modeling International.*, pages 165–171, 2003. doi:10.1109/SMI.2003.1199613.

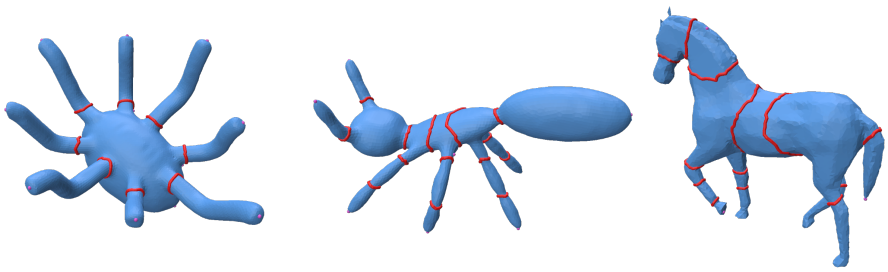


Figure 9 Inputs 3, 4, 5

19 Alfred Gray. *Tubes*. Birkhäuser Basel, 2nd edition, 2004. URL: http://dx.doi.org/10.1007/978-3-0348-7966-8, doi:10.1007/978-3-0348-7966-8.

20 Franck Hétroy and Dominique Attali. Detection of constrictions on closed polyhedral surfaces. In *5th Joint Eurographics - IEEE TCVG Symposium on Visualization, VisSym 2003, Grenoble, France, May 26-28, 2003*, pages 67–74. Eurographics Association, 2003. URL: https://doi.org/10.2312/VisSym/VisSym03/067-074, doi:10.2312/VISSYM/VISSYM03/067-074.

21 Franck Hétroy and Dominique Attali. From a closed piecewise geodesic to a constriction on a closed triangulated surface. In *11th Pacific Conf. Comp. Graph. and App.*, pages 394–398. IEEE Computer Society, 2003. Canmore, Canada, October 8-10, 2003. doi:10.1109/PCCGA.2003.1238282.

22 Hugh Howards, Michael Hutchings, and Frank Morgan. The Isoperimetric Problem on Surfaces. *The American Mathematical Monthly*, 106(5):430–439, 1999. Publisher: Mathematical Association of America. URL: https://www.jstor.org/stable/2589147, doi:10.2307/2589147.

23 F. Hétroy. Constriction Computation using Surface Curvature. In John Dingliana and Fabio Ganovelli, editors, *EG Short Presentations*. The Eurographics Association, 2005. doi:10.2312/egs.20051009.

24 Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8):649–658, September 2005. doi:10.1007/s00371-005-0344-9.

25 David Letscher and Jason Fritts. Image segmentation using topological persistence. In Walter G. Kropatsch, Martin Kampel, and Allan Hanbury, editors, *Computer Analysis of Images and Patterns*, pages 587–595, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

26 Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

27 Robert Osserman. The isoperimetric inequality. *Bulletin of the American Mathematical Society*, 84(6):1182–1238, 1978. URL: http://dx.doi.org/10.1090/S0002-9904-1978-14553-4, doi:10.1090/s0002-9904-1978-14553-4.

28 Nicholas Sharp, Keenan Crane, et al. Geometrycentral: A modern c++ library of data structures and algorithms for geometry processing. 2019.

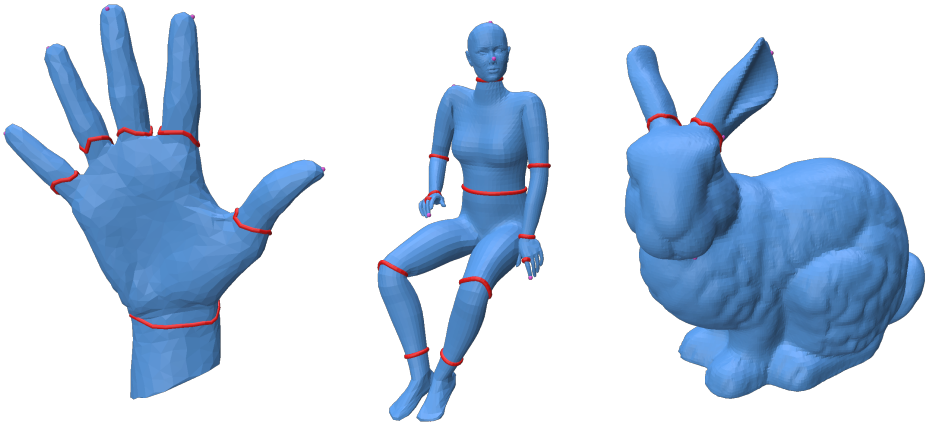


Figure 10 Inputs 6, 7, 8

29 Nicholas Sharp et al. Polyscope, 2019. www.polyscope.run.

30 Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. 3D Mesh Skeleton Extraction Using Topological and Geometrical Analyses. In *14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006)*, page s1poster, Tapei, Taiwan, October 2006. URL: https://hal.science/hal-00725576.

31 Shi-Qing Xin, Ying He, and Chi-Wing Fu. Efficiently Computing Exact Geodesic Loops within Finite Steps. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):879–889, June 2012. Conference Name: IEEE Transactions on Visualization and Computer Graphics. URL: https://ieeexplore.ieee.org/document/5928345, doi:10.1109/TVCG.2011.119.

32 Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24(1):1–27, January 2005. doi:10.1145/1037957.1037958.

33 Yinan Zhou and Zhiyong Huang. Decomposing polygon meshes by means of critical points. In *10th International Multimedia Modelling Conference, 2004. Proceedings.*, pages 187–195, January 2004. URL: https://ieeexplore.ieee.org/document/1264985, doi:10.1109/MULMM.2004.1264985.

		Runtime in ms			
Input	**# Salient Pts.**	**Salient Pts.**	**Cycles**	**Tightness**	**Total**
1	21	182	4252	398	4833
2	22	185	5428	413	6026
3	12	89	1352	129	1570
4	11	43	578	57	678
5	9	35	516	37	588
6	6	11	98	6	115
7	7	42	496	28	567
8	9	226	8151	230	8607

Table 2 Runtimes of our selected inputs. Our runtime is sensitive to the number of discovered points, which is also disclosed. $r = 20$ for salient point filtering. **Salient Pts.** is the time to discover the set of salient points and connect them into a skeleton. **Cycles** is the time for discovering every cycle along the skeleton. **Tightness**: the time for area computation, tightness computation, and cycle filtering.