

In the Search for Good Neck Cuts

Sariel Har-Peled ✉

Siebel School of Computing and Data Science, University of Illinois, Urbana, IL, USA

Sam Ruggiero ✉

Siebel School of Computing and Data Science, University of Illinois, Urbana, IL, USA

1 Abstract

We study the problem of finding neck-like features on a surface. Applications include mesh segmentation and shape analysis. We propose a new definition of a surface bottleneck — informally, a short cycle that bounds a large area on both sides. Inspired by the isoperimetric inequality, we formally define such optimal cuts, study their properties, and present a polynomial-time approximation algorithm under geometric regularity assumptions, together with a practical heuristic that works well on real meshes. 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

Funding *Sariel Har-Peled*: Partially supported by NSF AF award CCF-2317241.

Sam Ruggiero: Partially supported by NSF AF award CCF-2317241.

8 1 Introduction

Computing “good” cycles on surfaces is a well-studied problem [14, 6, 3, 5, 10, 15, 4, 2], including shortest geodesic cycles, non-contractible loops, handles, and more. 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 discussed in [35, 25]. Existing methods tend to rely on expensive topological preprocessing, 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 also present 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.

28 1.1 Background and prior work

Necks versus non-contractible loops. Finding neck-like features differs from finding the shortest non-contractible loops on a surface. As a reminder, a non-contractible loop on a surface corresponds to a cycle that cannot be homotoped to a point. Naturally, a neck-like loop might lie on an object that is topologically a ball (as is the case for examples shown in figures throughout this paper) – for example, on an hourglass, all the loops are contractible,



© Sariel Har-Peled Sam Ruggiero;

licensed under Creative Commons License CC-BY 4.0

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

XX:2 In the Search for Good Neck Cuts

34 yet it has a neck-like feature. In many high-genus objects, these non-contractible loops may
35 act as neck-like curves. However, there may be other non-contractible loops that do not lie
36 on a feature boundary, or contractible loops that are on neck-like features, which would not
37 be considered. Finding non-contractible cycles can be used to reduce the genus of a surface,
38 by cutting along them as shown in [14].

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

$$42 \quad \phi(S, \bar{S}) = \frac{|E(S, \bar{S})|}{\min(|S|, |\bar{S}|)}$$

43 is minimized. Unfortunately, the associated optimization problem is **NP-HARD**, and instead
44 one can use algebraic techniques to approximate it. One can use various heuristics inspired
45 by this observation to find good cuts. For example, Gotsman [17] noted the connection of
46 the Cheeger constant to the Laplacian of a surface mesh. Using this, he was able to detect
47 whether a small cut exists and how to partition the graph based on spectral embedding for
48 genus-0 models.

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

53 These techniques yield relatively slow algorithms, and because they do not work with the
54 geometry directly, the generated cuts are not locally optimal.

55 **Topological Methods.** Abdelrahman and Tong [1] computed neck-like features on meshes
56 by locating critical points in a volumetric mesh and generating cutting planes over the mesh
57 to isolate these loops. The main tool is identifying points that are 2-saddles of a Morse
58 function (generated by a distance function), as they are good seed locations for neck-like
59 features. This work extended Feng and Tong [15], who evaluated the persistent homology of
60 the mesh to locate neck-like loops.

61 Abdelrahman and Tong [1] achieve a speedup, compared to [15], by producing an initial
62 neck loop from the cutting plane, which is smoothed out via shortest loop evaluation. The
63 loops they generate are of good quality on all genus meshes. The main drawback, however, is
64 the need for a volumetric tetrahedral mesh, resulting in a substantial increase in vertex and
65 face density. Other topological approaches, such as the one by Dey *et al.* [12], often have the
66 same requirement of a volumetric mesh. Such a volumetric mesh can be substantially larger
67 than the input 2D surface, and generating it is itself challenging in some cases.

68 **Surface Methods.** Hétroy and Attali [22, 19, 20] compute geodesics on the surface and
69 slide them to generate tight constrictions (neck-like features). Earlier works relied on mesh
70 simplification to generate seed curves. However, in all cases, these algorithms rely on the
71 local properties of geodesics to find neck loops.

72 Specifically, Hétroy [22] approximates the mean curvature of the mesh in all locations
73 to find seed locations for constrictions. Then, the algorithm performs a local search from
74 these seed locations until the constrictions are minimized, and smooths and minimizes the
75 curve. Xin *et al.* [34] designed a fast algorithm to find shortest, exact geodesics on a model,

76 regardless of the quality of the input mesh. However, initial cutting loops must be specified
 77 in the input, as this algorithm does not discover loops from the input model directly.

78 Tierny *et al.* [33] use similar methods to our approach. However, when approximating
 79 constrictions, they use the concavity of the curve. This algorithm requires the Discrete
 80 Gaussian Curvature [26] at each point on the curve, which takes $O(n^2)$ time to compute.

81 1.2 Our approach

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

$$87 \frac{m(R)}{\|\partial R\|^2} \leq \frac{m(\mathcal{d})}{\|\partial \mathcal{d}\|^2} = \frac{\pi r^2}{(2\pi r)^2} = \frac{1}{4\pi},$$

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

94 **Computing tightness.** Unfortunately, computing (or even approximating) the tightest cycle
 95 on a surface appears to be hopeless in terms of efficient algorithms. Nevertheless, it provides
 96 us with an easily computable scoring function to compare cycles (i.e., the tighter, the better).
 97 There are cases where the optimal neck-cut is intuitively obvious, see [Figure 1](#). We thus
 98 investigate sufficient conditions under which we can efficiently approximate the optimal
 99 neck-cut. To this end, we first formally define tightness in [Section 2.1](#).

100 Specifically, for a loop, we look at the ratio between the area it encloses and its length
 101 squared (for a circle in the plane, this ratio is $1/4\pi$, as shown above). Clearly, the bigger
 102 the ratio, the more neck-like the cycle is. We formally define the underlying optimization
 103 problem in [Section 2.1](#).

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

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

116 In [Section 4](#), we present our practical algorithm. Our algorithm uses the salient points
 117 to define paths that the neck-like cycle must cross. The algorithm elegantly reduces to a

XX:4 In the Search for Good Neck Cuts

118 sequence of shortest-path computations with little preprocessing and focus on filtering cycles
119 of interest.

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

125 Numerous examples of the output of our algorithm are provided at [https://neckcut.](https://neckcut.space)
126 [space](https://neckcut.space) and discussed in [Section 5](#).

127 2 Isoperimetry, bottleneck cuts, and salient points

128 2.1 Isoperimetry and bottlenecks

129 **Isoperimetric problem on surfaces.** The isoperimetric problem asks one to find a plane
130 figure of maximum area, with a specified boundary length. This problem dates back to
131 antiquity, but a formal solution was not provided until the 19th century. It is known [7] that
132 circles, and in higher dimensions balls, are the optimal shapes. Even in the plane, proving it
133 was quite a challenge. For a planar closed curve σ , consider its *isoperimetric ratio*:

$$134 \quad \rho(\sigma) = \frac{\mathfrak{m}(\text{int}(\sigma))}{\|\sigma\|^2},$$

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

140 On a finite genus-zero surface in 3D, it is natural to seek a closed curve that is as short as
141 possible and that splits the surface into two “large” parts. As a concrete example, consider
142 the natural cycle at the base of a human finger: it does not partition the human mesh into
143 approximately equal parts, yet it is intuitively a good neck-cut.

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

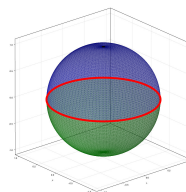
146 ► **Definition 3.** For a surface \mathcal{M} in \mathbb{R}^3 of genus-zero, and a region $\mathfrak{b} \subseteq \mathcal{M}$, the **tightness**
147 of \mathfrak{b} is the ratio

$$148 \quad \langle \mathfrak{b} \rangle = \frac{\min(\mathfrak{m}(\mathfrak{b}), \mathfrak{m}(\overline{\mathfrak{b}}))}{\|\partial\mathfrak{b}\|^2},$$

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

152 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
153 is the isoperimetric ratio. It is thus natural to ask for the patch $\mathfrak{b} \subseteq \mathcal{M}$ with maximum
154 tightness.

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



161 In general, proving that a specific patch $\mathfrak{b} \subseteq \mathcal{M}$ is the one realizing maximum tightness is
 162 a challenging problem, as the long history of the isoperimetric inequality testifies [27, 9, 21].
 163 A bottleneck curve on a surface should have the property that it is short, while enclosing a
 164 large area on both sides (e.g., a neck of an hourglass). Thus, our proxy for finding a good
 165 bottleneck curve is going to be computing curves on a given surface that have high tightness.

166 ▶ **Problem 5.** Given a surface \mathcal{M} , compute a region \mathfrak{b} with tightness $\langle\mathfrak{b}\rangle$ as large as possible.

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

171 2.2 Well-behaved surfaces

172 To provide some intuition for the approach used in Section 4, we discuss a few properties of
 173 a *well-behaved* surface.

174 **Slow-expansion on the surface.** We are interested in surfaces such that their measure (i.e.,
 175 area/volume) does not expand too quickly. To this end, given a set σ on \mathcal{M} , its *r-expansion*
 176 is the region

$$177 \quad \sigma \oplus r = \{p \in \mathcal{M} \mid d_{\mathcal{M}}(p, \sigma) \leq r\}, \quad (1)$$

178 where $d_{\mathcal{M}}$ is the geodesic distance along \mathcal{M} .

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

182 ▶ **Example 8.** In the plane, Steiner inequality [18] implies that for any curve σ we have
 183 $m(\sigma \oplus r) \leq \pi r^2 + 2r \|\sigma\|$ and $\|\partial(\sigma \oplus r)\| \leq 2\|\sigma\| + 2\pi r$. Thus, the plane is 2π -expanding.

184 ▶ **Definition 9.** A simple cycle σ is **contractible** if it can be homotoped to a single point.
 185 Given a region $\mathcal{R} \subseteq \mathcal{M}$ and a cycle $\sigma \subset \mathcal{R}$, the cycle is **\mathcal{R} -contractible** if it can be contracted
 186 to a point while staying inside \mathcal{R} .

187 If \mathcal{M} has genus-zero, any cycle σ on \mathcal{M} is contractible; however σ may still fail to be
 188 \mathcal{R} -contractible – for example, if \mathcal{R} is the result of taking \mathcal{M} and creating one puncture on
 189 each sides of σ . Intuitively, tight cycles are not locally contractible – one has to go far to be
 190 able to collapse them to a point.

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

XX:6 In the Search for Good Neck Cuts

196 ▶ **Remark 11.** Consider a region $\mathcal{R} \subseteq \mathcal{M}$, where \mathcal{M} is γ -expanding, such that any \mathcal{R} -
 197 contractible cycle σ in \mathcal{R} , is $((\sigma \oplus r) \cap \mathcal{R})$ -contractible, where $r = \gamma \|\sigma\|$. Then, the
 198 γ -expansion implies that $m(\mathcal{R}_\sigma) \leq m(\sigma \oplus r) \leq \gamma(\|\sigma\| r + r^2) \leq 2\gamma^3 \|\sigma\|^2$. Namely, \mathcal{R} is
 199 $2\gamma^3$ -well-behaved.

200 2.3 Salient points to a bottleneck

201 In the following, we assume the given surface \mathcal{M} is triangulated, genus 0, and has a useful
 202 collar (i.e., α -tight for a “large” α). In addition, we assume \mathcal{M} is γ -expanding, where γ is
 203 some small constant. Let \mathcal{o} denote this optimal α -bottleneck of \mathcal{M} . The cycle \mathcal{o} breaks \mathcal{M}
 204 into two regions \mathcal{b} and $\bar{\mathcal{b}}$. Let $s(\mathcal{o}, \mathcal{b})$ be the point furthest away from \mathcal{o} in \mathcal{b} . Formally, we
 205 define

$$206 \quad s(\mathcal{o}, \mathcal{b}) = \arg \max_{p \in \mathcal{b}} d_{\mathcal{M}}(p, \mathcal{o}) \quad \text{where} \quad d_{\mathcal{M}}(p, \mathcal{o}) = \min_{q \in \mathcal{o}} d_{\mathcal{M}}(p, q).$$

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

208 ▶ **Lemma 12** (salient points are far). *For $s = s(\mathcal{o}, \mathcal{b})$, we have that $d_{\mathcal{M}}(s, \mathcal{o}) \geq \|\mathcal{o}\|$, if*
 209 $\alpha \geq 4\gamma$.

210 **Proof.** Let $\ell = \|\mathcal{o}\|$. Since \mathcal{o} is an α -bottleneck, we have that

$$211 \quad m(\mathcal{b}) \geq \alpha \|\mathcal{o}\|^2.$$

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

$$213 \quad m(\mathcal{b}) \leq m(\mathcal{o} \oplus r) \leq \gamma(\|\mathcal{o}\| r + r^2).$$

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

$$215 \quad \frac{\alpha}{\gamma} \|\mathcal{o}\|^2 \leq (\|\mathcal{o}\|/2 + r)^2 \quad \implies \quad r \geq \left(\sqrt{\frac{\alpha}{\gamma}} - \frac{1}{2} \right) \|\mathcal{o}\|$$

216 as $\alpha/\gamma \geq 4$. ◀

217 3 Approximating the optimal collar

218 3.1 Identifying the neck where the collar lies

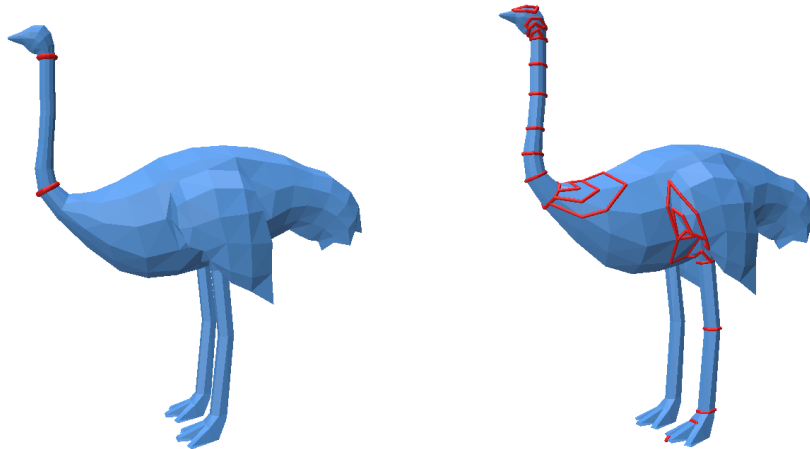
219 Consider the easy case where there is a good collar that is also stable: one can slide it up
 220 and down the “neck” and the quality of the collar remains relatively the same, see [Figure 1](#).

223 ▶ **Definition 13.** *Let κ be a “long” shortest path on \mathcal{M} with endpoints s and t . For every*
 224 *point $p \in \kappa$, cut \mathcal{M} along κ and consider the shortest path $\mathcal{C}_\kappa(p)$ from p to its copy on the*
 225 *other side of the cut. The closed curve $\mathcal{C}_\kappa(p)$ is a **lasso** if*

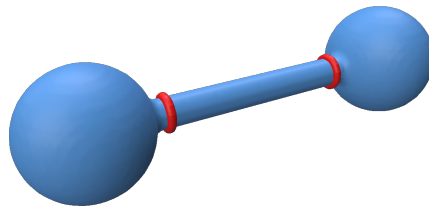
$$226 \quad \|\mathcal{C}_\kappa(p)\| < \max(d_{\mathcal{M}}(p, s), d_{\mathcal{M}}(p, t)),$$

227 *and is denoted by $\mathcal{C}_\kappa(p)$.*

228 Intuitively, a lasso is a “short” closed curve connecting p to itself, which is shorter than
 229 going from p to itself by going along the cut formed by κ .



221 ■ **Figure 1** Left: a long neck with a stable collar. Right: every lasso $\mathcal{O}_\kappa(p)$ generated as p ranges
 222 over the shortest path from the beak to the left foot.



235 ■ **Figure 2** A dumbbell object with the “neck” identified by red curves.

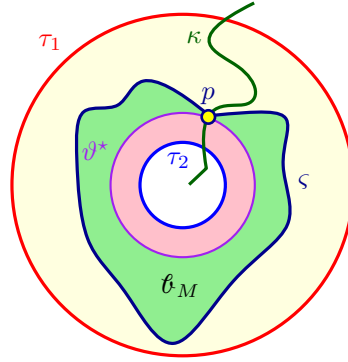
230 ► **Example 14.** Let \mathcal{M} be the surface of the following solid – connect two large disjoint balls
 231 by a long, thin cylinder (i.e., a dumbbell) – see **Figure 2**. Consider the cylinder portion of
 232 the surface – it forms a natural neck, and let \mathcal{R} denote it. Any curve going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

236 ► **Observation 15.** *Two lassos defined using the same base path κ cannot cross each other.*

237 Consider two lassos τ_i, τ_j induced by points $p_i, p_j \in \kappa$. Suppose τ_j crosses τ_i . Because
 238 τ_j and τ_i are geodesic cycles, one could shorten τ_j by using the portion of τ_i crossed by τ_j ,
 239 which is a contradiction. This holds for any two geodesic cycles using the same base path on
 240 \mathcal{M} , regardless if the lasso condition holds.

241 ► **Definition 16.** *Consider two lassos τ_1, τ_2 defined using a base path (which is a shortest
 242 path) κ . The **neck** $\mathcal{N} = \mathcal{N}(\tau_1, \tau_2)$ is the region on the surface \mathcal{M} lying between τ_1 and τ_2 .
 243 For a fixed $\beta > 0$, such a region is a **β -neck** if it is β -well-behaved.*

246 ► **Lemma 17.** *Let s, t be two points on \mathcal{M} , and let κ be the shortest path connecting them.
 247 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.*



244 ■ **Figure 3** A path κ that intersects ϑ^* and ς at p along a neck defined by τ_1 and τ_2 . ς only
 245 intersects ϑ^* at p without crossing.

248 Furthermore, assume that the optimal collar ϑ^* is contained in \mathcal{N} , with tightness $\beta \geq 8\alpha$.
 249 Then, there exists a point $p \in \kappa \cap \mathcal{N}$, such that its lasso $\varsigma = \mathcal{O}_\kappa(p)$ is $\beta(1 - \frac{4\alpha}{\beta})$ -tight.

250 **Proof.** Let p be any point in $\kappa \cap \vartheta^*$ as the base point for the construction, and let $\varsigma = \mathcal{O}_\kappa(p)$
 251 be the associated lasso.

252 Assume, for now, that the lasso ς only intersects ϑ^* at p (as in **Figure 3**). In that case, ς
 253 and ϑ^* are homotopic, and let \mathcal{C}_i be the area in between them. By the α -well-behavedness
 254 of \mathcal{N} , and since $\partial\mathcal{C}_i$ is $\vartheta^* \cup \varsigma$ (and is \mathcal{N} -contractible), we have that

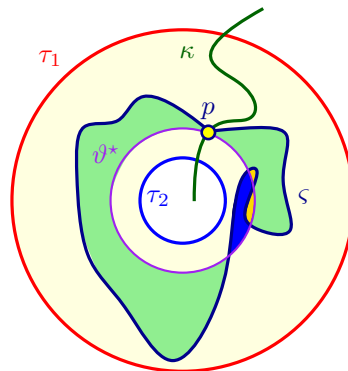
255
$$m(\mathcal{C}_i) \leq \alpha(\|\vartheta^*\| + \|\varsigma\|)^2 \leq 4\alpha \|\vartheta^*\|^2,$$

256 as $\|\varsigma\| \leq \|\vartheta^*\|$ – indeed, ϑ^* is a candidate for the shortest path connecting p to itself “around”
 257 κ , but ς is the shortest one.

258 For a closed curve η , let \mathcal{C}_η and $\bar{\mathcal{C}}_\eta$ be the two parts of \mathcal{M} bounded by η . Observe that

259
$$B = \min(m(\mathcal{C}_\varsigma), m(\bar{\mathcal{C}}_\varsigma)) \geq \min(m(\mathcal{C}_{\vartheta^*}), m(\bar{\mathcal{C}}_{\vartheta^*})) - m(\mathcal{C}_i).$$

260 The tightness of ς is thus $\langle \varsigma \rangle = \frac{B}{\|\varsigma\|^2} \geq \frac{\min(m(\mathcal{C}), m(\bar{\mathcal{C}})) - m(\mathcal{C}_i)}{\|\vartheta^*\|^2} \geq \langle \vartheta^* \rangle - 4\alpha.$



261 ■ **Figure 4** A path κ that intersects ϑ^* and ς at p along a neck defined by τ_1 and τ_2 . ς crosses ϑ^*
 262 multiple times, forming a series of contractible patches.

263 The slightly harder case is when ς and ϑ^* intersect at more than one point. In that
 264 case, the set $\varsigma \cup \vartheta^*$ forms an arrangement (see **Figure 4**) – the “inner” region/face denoted

265 by \mathcal{C}_s and the outer face denoted by \mathcal{C}_t . So consider all the other faces f_1, \dots, f_k in this
 266 arrangement. These faces are all contractible disks. Let ℓ_i be the boundary of the i th face,
 267 for $i = 1, \dots, k$. Observe that every edge e of ζ or ϑ^* contributes at most $2\|e\|$ to the total
 268 lengths of these face boundaries. Thus, we have $\sum_{i=1}^k \ell_i \leq 2(\|\zeta\| + \|\vartheta^*\|) \leq 2\|\vartheta^*\|$. Setting
 269 $\mathcal{C}_i = (\mathcal{C}_{\vartheta^*} \setminus \mathcal{C}_\zeta) \cup (\mathcal{C}_\zeta \setminus \mathcal{C}_{\vartheta^*})$ to be region that is the symmetric difference between $\mathcal{C}_{\vartheta^*}$ and
 270 \mathcal{C}_ζ , we have

$$271 \quad m(\mathcal{C}_i) \leq \sum_{t=1}^k m(f_t) \leq \sum_{t=1}^k \alpha \ell_t^2 = \alpha \sum_{t=1}^k \ell_t^2 \leq \alpha \left(\sum_{t=1}^k \ell_t \right)^2 \leq 4\alpha \|\vartheta^*\|^2.$$

272 The claim now follows from the argument above, and observing that $\langle \zeta \rangle \geq \langle \vartheta^* \rangle - 4\alpha =$
 273 $\beta \left(1 - \frac{4\alpha}{\beta}\right)$. ◀

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

277 3.2 The algorithm for computing the collar

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

281 We outline here the basic idea – our purpose is to present a polynomial-time approximation
 282 algorithm. To this end, we guess the points s and t of Lemma 17, and compute the shortest
 283 path κ between them. Next, we guess the two points, $x, x' \in \kappa$, defining the lassos bounding
 284 the optimal collar ϑ^* , and we compute these two lassos. We compute the region \mathcal{N} bounded
 285 in between τ_1 and τ_2 , and verify that it indeed has the topology of a sleeve (for example, by
 286 computing its Euler characteristic, and verifying that τ_1 and τ_2 cover all the boundary edges
 287 of this patch). Now, compute the shortest path around the neck for each vertex $v \in \kappa \cap \mathcal{N}$,
 288 and explicitly determine its tightness. The maximum one found is the desired approximation.
 289 If the model has size n , the running time of this algorithm is $O(n^6 \log n)$. This yields the
 290 following theorem.

291 ▶ **Theorem 19.** *Let \mathcal{M} be a triangulated surface in 3D with genus 0 and n vertices. Assume*
 292 *the optimal collar ϑ^* on \mathcal{M} lies on an α -neck \mathcal{N} that is induced by a shortest path κ , and two*
 293 *lassos τ_1, τ_2 (see Lemma 17). Furthermore, the tightness $\langle \vartheta^* \rangle \geq 8\alpha$. Then, one can compute,*
 294 *in $O(n^6 \log n)$ time, a closed curve ζ such that $\langle \zeta \rangle \geq \langle \vartheta^* \rangle - 4\alpha \geq \langle \vartheta^* \rangle / 2$.*

295 **Proof.** For every guess of s, t, x, x' , we can compute up to $O(n)$ lassos on \mathcal{M} , resulting in a
 296 total runtime of $O(n^6 \log n)$. By Lemma 17, one of these lassos ζ will be at least $\beta - 4\alpha$ -tight,
 297 where $\beta = \langle \vartheta^* \rangle$, thus $\langle \zeta \rangle \geq \langle \vartheta^* \rangle / 2$. ◀

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

303 4 A practical algorithm

304 We present here a practical algorithm that utilizes a few heuristics to locate optimal bottleneck
 305 curves while maintaining fast performance. This algorithm is an approximation and may not
 306 find all optimal curves. We assume the input is a surface model of genus-zero. [Theorem 19](#)
 307 suggests searching over every pair (s, t) to locate all possible necks. In practice, however, 3D
 308 models will have distinct features, which will guide our choices for (s, t) pairs to search for
 309 optimal lassos.

310 In the first stage, the algorithm locates salient points that lie on the tips of features. In
 311 the second stage, the algorithm takes paths between salient points and searches for bottleneck
 312 curves. We next describe these in detail. In [Section 5](#), we discuss how this algorithm performs
 313 on various real-world inputs.

314 4.1 Computing salient points

315 We aim to identify distant pairs of *salient points* from which to search for bottleneck curves.
 316 These points typically correspond to the tips of features on a mesh (a spike, finger, or head),
 317 but they do not necessarily lie on the convex hull. If our model has a good collar, [Lemma 12](#)
 318 tells us that looking along the paths between these salient points will locate the collar.
 319 Finding exact salient points described in [Section 2.3](#) is expensive; we use a standard two-pass
 320 shortest-path heuristic that locates good approximations of these points, though the result
 321 is sensitive to the initial seed. Previous work has focused on identifying salient points and
 322 utilizing them in mesh decomposition [36]. Other methods have also utilized salient points
 323 and surface methods to perform mesh decomposition [13, 16, 24, 11]. We use shortest path
 324 algorithms to locate these salient points, which act as reasonable estimates for the exact
 325 salient points that would be used to find bottleneck cuts.

326 We emphasize that in the following, we treat the mesh as a graph, where each edge has
 327 weight equal to the Euclidean distance between its endpoints.

328 4.1.0.1 Stage I: Salient point via approximate diameter.

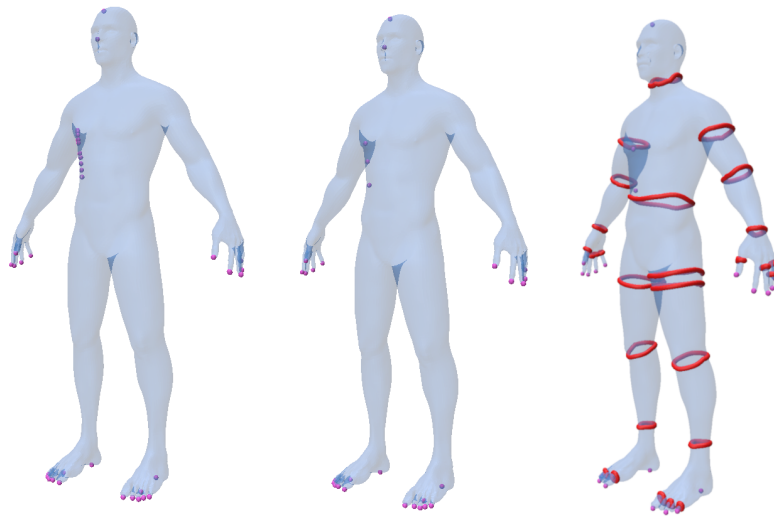
329 The algorithm first computes a point on the mesh that is part of a 2-approximation of the
 330 diameter of the mesh:

- 331 (I) The algorithm picks an arbitrary vertex s on the graph.
- 332 (II) The algorithm computes the shortest-path tree T_s from s using Dijkstra. Let u be the
 333 leaf with the maximum distance from s .
- 334 (III) The algorithm computes the shortest-path tree T_u from u , again using Dijkstra. Let v be
 335 the leaf with maximum distance from u . The pair (u, v) gives a 2-approximation to the
 336 diameter of the mesh when distances are measured along edge-paths.

337 A vertex x is a *salient point* if it is a local maximum of its neighbors with respect to
 338 distance to u (ties broken arbitrarily). That is,

$$339 \quad \forall y \in \Gamma(x) \quad d_{\mathcal{M}}(u, y) \leq d_{\mathcal{M}}(u, x), \quad (2)$$

340 where $\Gamma(x)$ is the *neighborhood* of x (i.e., set of all vertices adjacent to x). Because v is the
 341 furthest point from u and \mathcal{M} is connected, every neighbor of v has a strictly smaller distance
 342 to u , thus v is itself a salient point.



354 ■ **Figure 5** The salient points of a human mesh, with no filtering (left), $r = 10$ (middle), and $r = 20$
 355 (right). The bottleneck curves are shown in the right mesh.

343 4.1.0.2 Stage II: Collecting candidates for salient points.

344 Let C denote the set of candidate points. For every leaf x on T_u , we check all its neighbors
 345 on the mesh and mark it as a salient point if it is a local maximum of the distance function
 346 from u . That is, the algorithm checks if x complies with Eq. (2) and adds it to C .

347 4.1.0.3 Stage III: Filtering the candidates.

348 Depending on mesh quality, further filtering of salient points may be necessary; noisy surfaces
 349 can produce several local maxima close together. We eliminate any salient point that is
 350 within a user-selected hop distance r of another salient point. To this end, the algorithm
 351 performs an r -depth breadth-first search from each point of C , removing any point that is
 352 not a local maximum within its r -hop neighborhood. In practice r is a small constant, which
 353 suffices to handle small perturbations. See Figure 5.

356 ► **Observation 21.** For constant r , the algorithm above runs in $O(n \log n)$ time, assuming
 357 the maximum vertex degree in the mesh is bounded. Indeed, it involves running the Dijkstra
 358 algorithm several times, and performing local BFS of depth at most r for each candidate
 359 salient point.

360 4.2 Finding bottleneck curves

361 4.2.1 Generating candidate neck-cuts from a shortest path

362 Consider a shortest path κ . The algorithm runs a shortest-path algorithm from some vertex
 363 $p \in \kappa$ to itself, without crossing κ (except at p itself). That is, the algorithm “cuts” the
 364 surface along κ , computing shortest paths “around” κ from p to its copy on the other side of
 365 κ . This path is a geodesic (with respect to κ) cycle that includes p . This process generates a
 366 sequence of cycles Π , ordered along κ by their base vertex p . The task is to select cycles with
 367 a local maximum tightness. Computing the tightness requires computing the area bounded

368 by each cycle $\xi \in \Pi$, and selecting local maxima within Π . On genus-zero objects, we can
 369 compute the area between every adjacent pair of cycles (i.e. $\xi_i, \xi_{i+1} \in \Pi$ for each i), by doing
 370 BFS on the area in between the two curves, and then use prefix sums to compute the area
 371 bounded by one cycle efficiently. Non-crossing is guaranteed by [Observation 15](#). A cycle ξ
 372 whose tightness is a local maximum along its base path is reported as a candidate bottleneck.

373 Cycles near the endpoints of κ may be of low quality. Likewise, cycles that bound a
 374 small amount of area, while also being short, can lead to other cycles that appear to be local
 375 maxima among their neighbors. To that end, we refer to the calculation in [Example 4](#), and
 376 filter out any cycle with a computed tightness less than $\frac{1}{2\pi}$. Cycles with very low tightness
 377 are not representative of a large enough bounded area, and thus do not make good bottleneck
 378 curves.

379 4.2.2 Computing a skeleton: A small set of shortest paths

380 To apply the above procedure, we must supply it with a collection of shortest paths. A
 381 natural but naive approach is to take every pair of salient points in C and compute the
 382 shortest path between them. To reduce the number of paths, we compute a “cheap” skeleton
 383 of the salient points. A natural candidate for such a skeleton is the Steiner tree of the points
 384 of C on \mathcal{M} , but unfortunately, computing the exact Steiner tree is **NP-HARD**. To avoid
 385 this, we use a heuristic to construct such a tree. The algorithm starts with the approximate
 386 diametrical pair (u, v) and connects them by their shortest path. The algorithm then adds
 387 the remaining points to the constructed graph iteratively, connecting the i th salient point
 388 to its nearest neighbor on the tree constructed so far, adding this connecting path to the
 389 collection of shortest-paths computed; their union is the computed skeleton. We then use
 390 these paths with the above procedure to generate good neck-cuts.

391 ► **Remark 22.** The algorithm above offers only a $O(\log n)$ approximation [23]. A 2-approximation
 392 can be obtained with slightly more care. Let $C = \{s_1, \dots, s_k\}$. In iteration i , for
 393 $i = 2, \dots, k - 1$, the algorithm computes the closest point in $C \setminus C_i$ to $C_i = \{c_1, \dots, c_i\}$ via
 394 a single run of Dijkstra; call this point c_{i+1} . The algorithm then computes the shortest path
 395 from c_{i+1} to T_{i-1} (the tree after $i - 1$ iterations) and adds it to form T_i . This algorithm can
 396 be viewed as running Prim’s algorithm on the induced complete graph over C (where the
 397 weights are the shortest path distances between the corresponding points on the model). The
 398 constructed tree T_k is clearly no heavier than this MST, which in turn is a 2-approximation
 399 to the optimal Steiner tree. The running time of this algorithm is $O(kn \log n)$. We have not
 400 implemented this exact variant, since it does not seem to matter in practice.

401 4.3 Running time analysis

402 Computing the set of candidates takes $O(n \log n)$ time (computing the shortest-path tree
 403 plus r -depth local filtering at each leaf). Let $k = |C|$ and $|K|$ be the number of vertices in the
 404 skeleton. Computing the skeleton itself takes $O(kn \log n)$ time. The bulk of our time results
 405 from computing the cycles along the paths, which takes $O(|K|n \log n)$ time. In practice, we
 406 can expect the number of vertices in a shortest path to be $O(\sqrt{n})$ [32], and k to be small
 407 and independent of n [33] (see [Table 2](#)). Thus our skeleton size (as a collection of shortest
 408 paths) is $|K| = O(\sqrt{n})$ empirically.

409 There are additional optimizations in practice that can be employed that would further
 410 speed up the cycle searching. Mainly, computing the cycles along each path is not reliant on
 411 data from the other paths in K . With parallelism, each path can be computed independently,

412 thus bounding the runtime based on the number of available cores and the longest path
413 within the skeleton.

414 5 Evaluation

415 We implemented our algorithm in C++, using the Polyscope & Geometry Central [30, 29]
416 libraries. These libraries provide a halfedge data structure, with standard traversal algorithms.
417 All timings were measured on a single thread of an Intel i7-14700K 3.4GHz CPU. We tested our
418 algorithm on *Benchmark for 3D Mesh Segmentation Dataset* [8], along with additional meshes.
419 All of our results from this dataset, along with the code, can be viewed on meshcuts.space.
420 We have selected a few to feature in this paper, as shown in Table 1. We measured the
421 runtime of all genus-zero models, as shown in Figure 6. We also used the code from [1] for
422 comparison, using a smoothing level of 20 as in the original paper. However, we were unable
423 to get the cycle extraction stage to work using the author-provided code, thus that stage is
424 excluded from the reported runtime, and we defer to their paper for additional figures.

Input	Description	Faces
1	Public Domain: Human 1	48,918
2	MSB/Stanford: Armadillo	50,542
3	MSB: Octopus	28,248
4	MSB: Ant	13,696
5	MSB: Horse	11,072
6	MSB: Hand	3,026
7	MSB: Human 2	11,258
8	Stanford: Bunny	69,630

434 ■ **Table 1** Selected inputs from our testing. MSB models are from [8], Stanford models are from
435 [31].

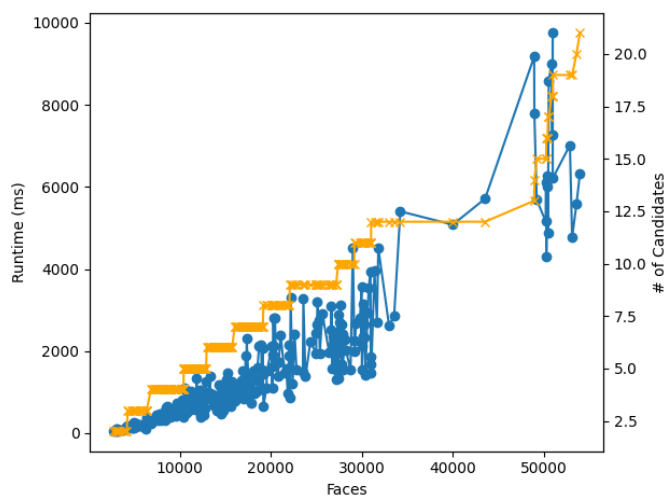
436 Our implementation still has some optimizations yet to be implemented, as described
437 in Section 4.3. The timings described in Table 2 are single-threaded operations. We lazily
438 compute the bounded area in this implementation, only running the prefix-sum method per
439 path, rather than the entire skeleton at once. The results from these models can be seen in
440 Figure 7, Figure 8, and Figure 9.

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

448 Additionally, we choose not to perform any mesh or cycle smoothing. One could pass the
449 output cycles from our algorithm to a geodesic refiner like [28], but this technique performs
450 internal refinement of the mesh to produce smooth geodesics.

471 6 Conclusions

472 We introduced an isoperimetric formulation of neck-like features: the *tightness* of a cycle as
473 a local isoperimetric ratio on a mesh. Based on this definition, we presented two algorithms.



451 ■ **Figure 6** Runtime of models from [8]. Runtime is plotted with *blue* dots. The number of
 452 candidates is plotted with *orange* crosses.

474 The first is a polynomial-time approximation algorithm for the optimal collar on well-behaved
 475 surfaces, which establishes that the problem is tractable under regularity assumptions. The
 476 second is a practical heuristic that reduces the whole pipeline to a small number of shortest-
 477 path computations on a skeleton of salient points, followed by per-path cycle enumeration
 478 and tightness filtering. Empirically, the resulting algorithm runs in sub-quadratic time.
 479 Our method has improvements over previous work [33, 22, 1], without the machinery those
 480 approaches require.

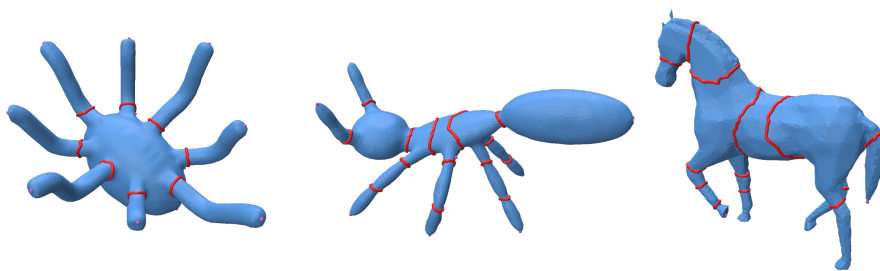
481 We view the simplicity of the practical algorithm as its main strength: the only primitives
 482 it needs are a halfedge mesh and a shortest path algorithm. This makes it straightforward to
 483 integrate into existing pipelines, and easy to extend.

484 **Limitations.** The area-based tightness computation in the practical algorithm relies on BFS
 485 between adjacent cycles and is currently formulated for genus-zero meshes; higher-genus
 486 surfaces would require either handle decomposition or a different area accounting scheme.
 487 We attempted our implementation on higher-genus surfaces, but the skeleton generation
 488 algorithm described would have to be modified to output a skeleton which resembles the
 489 object’s Reeb graph. However, computing the local patch around a candidate cycle on a
 490 high-genus object gave valid tightness quantities and identified bottleneck cycles.

491 **Future Work.** We plan to explore using this algorithm as a step in other applications. One
 492 possible direction is to use the bottleneck curves as the seed cycles for [34], rather than
 493 user-defined cutting planes. Additionally, the embarrassingly parallel nature of the cycle
 494 search makes this a good candidate for a real-time and dynamic implementation, as models
 495 are morphed and deformed.

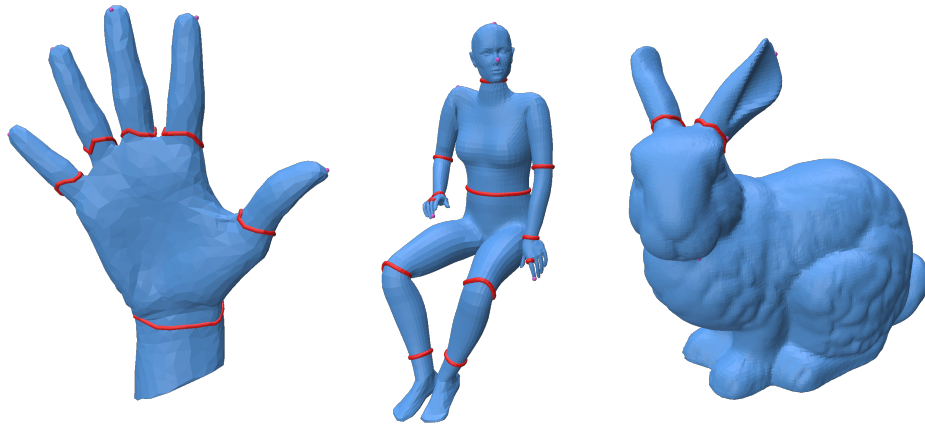


453 ■ **Figure 7** Inputs 1 & 2



454 ■ **Figure 8** Inputs 3, 4, 5





455 ■ **Figure 9** Inputs 6, 7, 8

		Runtime in ms				
Input	#	Salient	Cycles	Tightness	Total	FCNLF
1	21	182	4,252	398	4,833	176,420
2	22	185	5,428	413	6,026	137,937
3	12	89	1,352	129	1,570	58,669
4	11	43	578	57	678	22,669
5	9	35	516	37	588	21,880*
6	6	11	98	6	115	8,468
7	7	42	496	28	567	47,510*
8	9	226	8,151	230	8,607	292,390*

465 ■ **Table 2** Runtimes of the algorithm on selected inputs. Runtime is sensitive to the number of
 466 discovered points, also reported. Here, $r = 20$ is used for salient-point filtering. Column # is the
 467 number of salient points found. The *Salient* column is the time to discover the salient points and
 468 connect them into a skeleton; the *Cycles* column is the time for discovering cycles along the skeleton;
 469 the *Tightness* column is the time for area and tightness computation plus cycle filtering. FCNLF is
 470 the runtime for [1], * indicates a crash during saddle point selection.

496 ——— **References** ———

- 497 1 Hayam Abdelrahman and Yiyong Tong. Fast Computation of Neck-Like Features. *IEEE*
498 *Transactions on Visualization and Computer Graphics*, 29(12):5384–5393, December 2023.
499 Conference Name: IEEE Transactions on Visualization and Computer Graphics. URL:
500 <https://ieeexplore.ieee.org/document/9910018>, doi:10.1109/TVCG.2022.3211781.
- 501 2 Florian Beguet, Sandrine Lanquetin, and Romain Raffin. Flexible mesh segmentation via
502 Reeb graph representation of geometrical and topological features. *CoRR*, abs/2412.05335,
503 2025. arXiv:2412.05335, doi:10.48550/arXiv.2412.05335.
- 504 3 Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Reeb graphs for
505 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.
- 506 4 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding shortest non-trivial
507 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.
- 508 5 Sergio Cabello, Matt DeVos, Jeff Erickson, and Bojan Mohar. Finding one tight cycle. *ACM*
509 *Trans. Algorithms*, 6(4):61:1–61:13, 2010. doi:10.1145/1824777.1824781.
- 510 6 Sergio Cabello and Bojan Mohar. Finding shortest non-separating and non-contractible cycles
511 for topologically embedded graphs. *Discret. Comput. Geom.*, 37(2):213–235, 2007. URL:
512 <https://doi.org/10.1007/s00454-006-1292-5>, doi:10.1007/S00454-006-1292-5.
- 513 7 Lamberto Cesari. Rectifiable Curves and the Weierstrass Integral. *The American Mathematical*
514 *Monthly*, 65(7):485–500, 1958. Publisher: [Taylor & Francis, Ltd., Mathematical Association
515 of America]. URL: <https://www.jstor.org/stable/2308574>, doi:10.2307/2308574.
- 516 8 Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh
517 segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.
- 518 9 Andrew Cotton, David Freeman, Andrei Gnepp, Ting Ng, John Spivack, and Cara Yoder. The
519 isoperimetric problem on some singular surfaces. *Journal of the Australian Mathematical Society*,
520 78(2):167–197, April 2005. URL: [https://www.cambridge.org/core/product/identifier/
521 S1446788700008016/type/journal_article](https://www.cambridge.org/core/product/identifier/S1446788700008016/type/journal_article), doi:10.1017/S1446788700008016.
- 522 10 Tamal K. Dey, Fengtao Fan, and Yusu Wang. An efficient computation of handle and tunnel
523 loops via reeb graphs. *ACM Trans. Graph.*, 32(4):32:1–32:10, 2013. doi:10.1145/2461912.
524 2462017.
- 525 11 Tamal K. Dey, Joachim Giesen, and Samrat Goswami. Shape segmentation and matching with
526 flow discretization. In Frank Dehne, Jörg-Rüdiger Sack, and Michiel Smid, editors, *Algorithms*
527 *and Data Structures*, pages 25–36, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- 528 12 Tamal K. Dey, Kuiyu Li, Jian Sun, and David Cohen-Steiner. Computing geometry-aware
529 handle and tunnel loops in 3D models. *ACM Transactions on Graphics*, 27(3):1–9, August
530 2008. URL: <https://dl.acm.org/doi/10.1145/1360612.1360644>, doi:10.1145/1360612.
531 1360644.
- 532 13 Vladislav Dordiuik, Maksim Dzhigil, and Konstantin Ushenin. Surface Mesh Segmentation
533 Based on Geometry Features. In *2023 IEEE Ural-Siberian Conference on Biomedical*
534 *Engineering, Radioelectronics and Information Technology (USBREIT)*, pages 270–273,
535 May 2023. ISSN: 2769-3635. URL: <https://ieeexplore.ieee.org/document/10158888>,
536 doi:10.1109/USBREIT58508.2023.10158888.
- 537 14 Jeff Erickson and Sarel Har-Peled. Optimally Cutting a Surface into a Disk. *Discrete Comput.*
538 *Geom.*, 31(1):37–59, January 2004. doi:10.1007/s00454-003-2948-z.
- 539 15 Xin Feng and Yiyong Tong. Choking Loops on Surfaces. *IEEE Transactions on Visualization*
540 *and Computer Graphics*, 19(8):1298–1306, August 2013. Conference Name: IEEE Transactions
541 on Visualization and Computer Graphics. URL: [https://ieeexplore.ieee.org/document/
542 6409845](https://ieeexplore.ieee.org/document/6409845), doi:10.1109/TVCG.2013.9.
- 543 16 Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3D models.
544 *Computers & Graphics*, 33(3):262–269, June 2009. URL: [https://www.sciencedirect.com/
545 science/article/pii/S0097849309000454](https://www.sciencedirect.com/science/article/pii/S0097849309000454), doi:10.1016/j.cag.2009.03.010.
- 546
- 547

- 548 17 C. Gotsman. On graph partitioning, spectral analysis, and digital mesh processing. In *2003*
549 *Shape Modeling International.*, pages 165–171, 2003. doi:10.1109/SMI.2003.1199613.
- 550 18 Alfred Gray. *Tubes*. Birkhäuser Basel, 2nd edition, 2004. URL: [http://dx.doi.org/10.1007/](http://dx.doi.org/10.1007/978-3-0348-7966-8)
551 [978-3-0348-7966-8](http://dx.doi.org/10.1007/978-3-0348-7966-8), doi:10.1007/978-3-0348-7966-8.
- 552 19 Franck Hétroy and Dominique Attali. Detection of constrictions on closed polyhedral surfaces.
553 In *5th Joint Eurographics - IEEE TCVG Symposium on Visualization, VisSym 2003, Grenoble,*
554 *France, May 26-28, 2003*, pages 67–74. Eurographics Association, 2003. URL: [https://doi.](https://doi.org/10.2312/VisSym/VisSym03/067-074)
555 [org/10.2312/VisSym/VisSym03/067-074](https://doi.org/10.2312/VisSym/VisSym03/067-074), doi:10.2312/VISSYM/VISSYM03/067-074.
- 556 20 Franck Hétroy and Dominique Attali. From a closed piecewise geodesic to a constriction on a
557 closed triangulated surface. In *11th Pacific Conf. Comp. Graph. and App.*, pages 394–398.
558 IEEE Computer Society, 2003. Canmore, Canada, October 8-10, 2003. doi:10.1109/PCCGA.
559 2003.1238282.
- 560 21 Hugh Howards, Michael Hutchings, and Frank Morgan. The Isoperimetric Problem
561 on Surfaces. *The American Mathematical Monthly*, 106(5):430–439, 1999. Publisher:
562 Mathematical Association of America. URL: <https://www.jstor.org/stable/2589147>,
563 doi:10.2307/2589147.
- 564 22 F. Hétroy. Constriction Computation using Surface Curvature. In John Dingliana and
565 Fabio Ganovelli, editors, *EG Short Presentations*. The Eurographics Association, 2005. doi:
566 10.2312/egs.20051009.
- 567 23 Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM Journal*
568 *on Discrete Mathematics*, 4(3):369–384, 1991. arXiv:<https://doi.org/10.1137/0404033>,
569 doi:10.1137/0404033.
- 570 24 Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and
571 core extraction. *The Visual Computer*, 21(8):649–658, September 2005. doi:10.1007/
572 s00371-005-0344-9.
- 573 25 David Letscher and Jason Fritts. Image segmentation using topological persistence. In
574 Walter G. Kropatsch, Martin Kampel, and Allan Hanbury, editors, *Computer Analysis of*
575 *Images and Patterns*, pages 587–595, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- 576 26 Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-
577 geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier,
578 editors, *Visualization and Mathematics III*, pages 35–57, Berlin, Heidelberg, 2003. Springer
579 Berlin Heidelberg.
- 580 27 Robert Osserman. The isoperimetric inequality. *Bulletin of the American Mathematical Society*,
581 84(6):1182–1238, 1978. URL: <http://dx.doi.org/10.1090/S0002-9904-1978-14553-4>, doi:
582 10.1090/s0002-9904-1978-14553-4.
- 583 28 Nicholas Sharp and Keenan Crane. You can find geodesic paths in triangle meshes by just
584 flipping edges. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.
- 585 29 Nicholas Sharp, Keenan Crane, et al. Geometrycentral: A modern c++ library of data
586 structures and algorithms for geometry processing. 2019.
- 587 30 Nicholas Sharp et al. Polyscope, 2019. www.polyscope.run.
- 588 31 Stanford Computer Graphics Laboratory. The Stanford 3d scanning repository. [https:](https://graphics.stanford.edu/data/3Dscanrep/)
589 [//graphics.stanford.edu/data/3Dscanrep/](https://graphics.stanford.edu/data/3Dscanrep/), 2014. Accessed: December 2025.
- 590 32 Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe.
591 Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, July
592 2005. URL: <https://doi-org.proxy2.library.illinois.edu/10.1145/1073204.1073228>,
593 doi:10.1145/1073204.1073228.
- 594 33 Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. 3D mesh skeleton extraction
595 using topological and geometrical analyses. In *14th Pacific Conf. Comp. Graph. App.*, page
596 s1poster, Tapei, Taiwan, October 2006. URL: <https://hal.science/hal-00725576>.
- 597 34 Shi-Qing Xin, Ying He, and Chi-Wing Fu. Efficiently Computing Exact Geodesic Loops within
598 Finite Steps. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):879–889,

- 599 June 2012. Conference Name: IEEE Transactions on Visualization and Computer Graphics.
600 URL: <https://ieeexplore.ieee.org/document/5928345>, doi:10.1109/TVCG.2011.119.
- 601 35 Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization
602 and texture mapping. *ACM Trans. Graph.*, 24(1):1–27, January 2005. doi:10.1145/1037957.
603 1037958.
- 604 36 Yinan Zhou and Zhiyong Huang. Decomposing polygon meshes by means of critical
605 points. In *10th International Multimedia Modelling Conference, 2004. Proceedings.*, pages
606 187–195, January 2004. URL: <https://ieeexplore.ieee.org/document/1264985>, doi:
607 10.1109/MULMM.2004.1264985.