Versatile Curve Design by Level Set with Quadratic Convergence

Xiaohu Zhang, Shuang Wu, Jiong Chen, Yao Jin, Hujun Bao, Jin Huang

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365

Versatile Curve Design by Level Set with Quadratic Convergence

Xiaohu Zhang, Shuang Wu, Jiong Chen, Yao Jin, Hujun Bao, and Jin Huang

Abstract— Many 3D mesh processing tasks revolve around generating and manipulating curves on surface meshes. While it is intuitive to explicitly model these curves using mesh edges or parametric curves in the ambient space, these methods often suffer from numerical instability or inaccuracy due to the projection operation. Another natural strategy is to adapt spline based tools, these methods are quite fast but are hard to be extended to more versatile constraints and need heavy manual interactions. In this paper, we present an efficient and versatile approach to curve design based on an implicit representation known as the level set. While previous works have explored the use of the level set to generate curves with minimal length, they typically have limitations in accommodating additional conditions for rich and robust control. To address these challenges, we formulate curve editing with constraints like smoothness, interpolation, tangent control, *etc*., via a level set based variational problem by constraining the values or derivatives of the level set function. However, the widely used gradient flow strategy converges very slowly for this complicated variational problem compared to the classical geodesic one. Thus, we propose to solve it via Newton's method enhanced by local Hessian correction and a trust-region strategy. As a result, our method not only enables versatile control, but also excels in terms of performance due to nearly quadratic convergence and almost linear complexity in each iteration via narrow band acceleration. In practice, these advantages effectively benefit various applications, such as interactive curve manipulation, boundary smoothing for surface segmentation and path planning with obstacles as demonstrated.

Index Terms—Level Set, Curve, Mesh, Interactive Editing

1 Introduction

Drawing curves on a discrete surface plays an important role in many geometry processing tasks, particularly when it is necessary to partition an object into several components for further analysis. However, generating an ideal curve tailored to a specific task can pose considerable technical challenges. These challenges often arise from strong constraints imposed on curve geometry, which can be hard to precisely fulfill due to the choice of curve representation and associated numerical algorithms.

Non-variational methods([45, 46, 53]) can construct curves on mesh easily without solving complicated optimization problems, but they are not friendly to automatically impose various constraints(*e.g*. customized smoothness, avoiding obstacles, *etc*.) and require heavy user interactions. Thus, like many existing techniques [27, 32, 68], we resort to a variational framework which solves an optimization problem to get the curve fulfilling special constraints automatically.

The representation of the curve on the mesh has strong influences on the resulting optimization problem. Explicit representations, whether intrinsic (line segments exactly on mesh) [39] or extrinsic [32] (projecting a 3D curve to the target mesh), are intuitive and common choices. However, such representations often introduce numerical instability, inaccuracy and potential numerical failure because of frequent node adapting, tiny line segments or risky projection around concave regions for complicated optimization problems. In contrast, implicit representations naturally keep the curve on the surface by construction. One specific implicit representation, known as the "level set" method [52], adopts a dynamic or evolving view of curves or interfaces via a Hamilton-Jacobi-based flow, i.e., the gradient of the objective function. This technique has been widely used for computing geodesic lines/loops [65, 69], but has never been applied to curve editing with **various controls**.

In this paper, we show the efficiency of using level set functions for some curve editing tasks. We first propose a level set based variational method which incorporates interpolation, tangent control, *etc*., and then apply Willmore energy [5, 56] based smoothness term along with a signed distance regularization [36]. By virtue of the variational framework, we can even control the shape of the curve by introducing constraints that are not on the curve itself (*e.g*., avoiding some obstacles).

The versatility rooted in the flexible formulation comes with the challenge of numerical optimization. Conventional gradient-flow (i.e., a first order optimization strategy) leads to slow convergence, and renders the method impractical. Though a specifically designed gradient-flow method ([41, 65, 69]) can be efficient for geodesic lines associated with the vanilla minimal length energy, applying it to such a complicated optimization problem is not a trivial task. Thus, we explore a Newton-based method (i.e., a second-order optimization strategy) to evolve the curve. To make it possible, we give detailed derivatives on triangular meshes. By applying the local Hessian correction and trust-region strategy, the Newton's method results in much better convergence and stability than gradient flow. Along with a carefully designed narrow band scheme, the complexity of each iteration scales nearly linearly in the length of the current curve. The overall numerical approach proposed in this paper improves the efficiency of level set based curved design for interactive curve editing tasks and various applications involving high-resolution meshes.

The contributions of the work can be summarized as:

- the first level set based curve design on mesh with various controls, *e.g*., interpolation, smoothness, tangent, obstacles *etc*.,
- the discretization and derivatives of the terms on triangular mesh,
- and a reliable and efficient numerical method for practical usage.

1.1 Related work

In the following parts, we first introduce curve design on analytical Riemannian surface, and followed by polygonal (triangular) meshes, either explicit (i.e., a polyline in 3D space) or implicit (i.e., zero-contour of a scalar field).

Curve on Analytical Surface Curve design on parametric surface is a long-standing topic in CAGD with much work published. They mainly focused on interpolating/fitting time-labeled landmarks on analytical Riemannian surface, such as Euclidean plane(for validation),

- *Xiaohu Zhang, Shuang Wu, Hujun Bao, and Jin Huang are with Zhejiang University. Email: xiaohuzhang@zju.edu.cn, shuangwu2002@foxmail.com, {bao\|hj}@cad.zju.edu.cn*
- *Jiong Chen is with INRIA. Email: jiong.chen@inria.fr.*
- *Yao Jin is with Zhejiang Sci-Tech University. Email: jinyao@zstu.edu.cn.*
- *Xiaohu Zhang and Shuang Wu are co-first authors.*
- *Jin Huang is the corresponding author.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

2-sphere, rotation group and shape manifold. Gousenbourger et al. [23] first propose a framework to interpolate a C^1 continuous splines on Riemannian manifold by minimizing their mean squared acceleration; then, the suboptimal of its velocity(even for Euclidean case) was improved by [2]. In [22], they proposed a blended cubic spline method to extend the interpolation problem to fitting a C^1 curve by using a parameter λ to control the trade-off between "straight enough" and "closely enough". For ease of use, the only knowledge required from the manifold is Riemannian exponentials and logarithms. Bergmann et al. [4] proposed a variational scheme by approximating the acceleration using discrete squared second order derivative along the curve to fit a smooth spline; a closed-form, numerically stable and efficient algorithm to compute the gradient is derived, and this method struck a balance between a data proximity constraint and a smoothing regularization constraint.

In brief, although these methods are efficient and robust, they are not directly applicable and hard to be extended to discrete polygon meshes, which are widely used in geometry processing applications. Our work also use a variational framework with smoothness, interpolation/fitting constraints, but we extend it to more general polygon cases and introduce versatile controls, *e.g.*, tangent and obstacles *etc*. We also provide a reliable and efficient discretization and numerical method for practical usage.

Explicit Curve Design These methods represent a curve by a sequence of points in 3D space, and the key difficulty is to maintain such a curve lying on the surface mesh, while satisfying some other user controls, *e.g.*, smoothness, interpolation, *etc*.

An attractive and practical way to design curves on mesh is to extend Bézier curves on Euclidean space to manifold. The key idea is how to construct weighted Riemannian mass center (RCM) of control points properly. One possible way is to embed the mesh to higher-dimensional Euclidean space where the Euclidean distance can be computed in a closed-form, and the approximated RCM is projected back to the mesh surface [53]. The approximation in this method improves the efficiency but leads to artifacts from the embedding and projection. To achieve better accuracy, Mancinelli et al. [46] optimized the Riemannian mass center by tracing the Newton direction of geodesic distance field precomputed from a set of source control points. They get more accurate results than [53] with slightly lower speed. For both RCM based methods [46, 53], they share the same risk of failure when control points are too sparse. To avoid this robustness issue, Mancinelli et al. [45] proposed robust subdivision based methods. When interpolation curve is desired, rational splines can be used to fulfil it [46]. In general, these Bézier based methods are widely used for its excellent performance and convenience. However, if the curve is required to satisfy more constraints at the same time, *e.g.*, interpolation, smoothness, tangent control and avoiding certain obstacles *etc*., it is hard to be achieved by these methods automatically.

Variational methods provide the opportunity to impose various constraints. Some of them adopt the intrinsic strategy, which discretizes the curve on mesh as a sequence of points on mesh edges. Some intrinsic methods iteratively evolve an initial curve composed of a sequence of points on mesh edges by certain objective functional measuring smoothness [31, 34] or shortness [39]. The nodes will be added or removed adaptively, so that the edges in the polyline could be very short, which would bring difficulties in designing an efficient and robust numerical method for flexible optimization problem, *e.g.*, reliably computing the derivatives along the polyline when segments are nearly degenerate [39]. Other intrinsic methods like [35] simplify the problem into a 2D one through parameterization. They usually involve the reprocessing of parameterization, and it is hard to avoid the influence coming from the distortion of the parameterization.

Another kind of strategy is extrinsic, which relaxes the manifold constraint and solves the optimization problem in the ambient space of the mesh. Such a setting can flexibly control the curve properties through augmenting various energy terms, but it needs extra effort to snap the curve to the mesh surface by a certain projection operator. Some of them directly project the curve nodes to the mesh surface during the iterations [27, 54, 68], which may lead to robustness and efficiency issues. Jin et al. [32] used a surrounding shell space to assist fast and robust projection, but the construction of high-quality shell is difficult. Xu et al. [67] recently designed B-spline curves on mesh with the similar framework [32]. It adopts the simplified shell that can be built robustly, but is not efficient for designing complicated curves. In contrast, our method designs curve implicitly and can overcome the mentioned disadvantages of these explicit methods while still being efficient.

Implicit Curve Design In contrast to explicit methods, implicit methods usually construct a special scalar field on the mesh and extract the certain level set (i.e., a contour) as the curve to be designed. Level set method proposed in [52] is one of such methods, and it has been widely used in many applications such as fluid simulations [38, 71], mesh fairing [16], Willmore flow [5, 56], image segmentation [11, 37, 50], surface reconstruction [57, 64, 72], shape modeling [6, 44, 49], path planning [42, 43] and optimal design [17, 63] *etc*. Many of the methods apply the idea of level set to 2D or 3D flat Euclidean space and polygon meshes [65]. Comprehensive surveys about level set can be found in [8, 21, 51]. Here we only review the most related ones, especially focusing on curve design on meshes.

To the best of our knowledge, most methods optimize the curves on a surface by level set target on shortest curve under a prescribed and constant length weight. The weights can be uniform over the whole surface or non-uniform [33], but it does not change during the optimization. It should be noticed that the problem of minimal surface [14, 62] shares the similarity in using constant weight. Evolving the curve by conventional gradient-flow, which is derived from the gradient of objective function, converges slowly and is often stuck at poor local minima. Using the semi-implicit integrator [65] can significantly improve the convergence. However, it is hard to extend this technique to dynamically weighted objective functions like Willmore flow involved in [5,56,60], which can be viewed as weighting the length (area in 3D) by square of dynamically changed mean curvature. For dynamic weight, simple gradient flow is even more difficult to converge. Some methods involving dynamic weighted problem (*e.g.*, [58, 73]) can be efficient, but their objective function includes a regularization of small (weighted) area bounded by the curve based on the prior of input images. It makes sense for applications like image segmentation or contouring a background signal. However, such a regularization is not a general purpose of curve design, and cannot be used in our tasks without introducing undesired side effects. Methods like [12, 12, 59, 60] also utilize such prior in gradient-descent, but the paper [3] shows that the convergence of gradient-descent is still poor.

Thus, we turn the eyes to second-order numerical methods, i.e. Newton-based methods. Indeed, such methods have been explored in [3, 7, 25, 26] for active contour of images. Though they show superior convergence than gradient flow, the Newton's method is still rarely used in many level set related techniques, possibly because the Hessian is hard to derive, even for images. In this paper, we give the derivatives on triangular meshes. We noticed that [3] modified classic Newton's method by replacing the inner product with a smoothed version (convolution with a Gaussian) in second order Taylor expansion. This usually helps to get a better Hessian (smaller condition number or turning an indefinite Hessian into the positive-definite one). Smoothing the gradient helps on noisy images, but brings the risk of smoothing a non-zero gradient into a nearly zero one. This may slow down the convergence or lead to a solution different from the true one. Instead of smoothing the gradient, we use element-wise symmetric positive projection to keep the total Hessian positive definite without introducing the above issue.

2 Level Set on Surface

Before introducing our method, we briefly recap the method of level set, and the most related work in this section.

In the level set method, one of the key components is to turn the interface integration into the volumetric one by the co-area formula [20]. As an example, the length of a curve $\Gamma = \phi^{-1}(0)$ on a surface M can be transformed into a surface integration:

$$\int_{\Gamma} d\Gamma = \int_{M} \delta(\phi)|\nabla\phi| dM. \tag{1}$$

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365

The Dirac Delta function δ plays an important role here that weighs all the point with nonzero ϕ by zero, and the co-area term $|\nabla\phi|$ comes from integration by substitution. For relatively simple objectives, the gradient can be computed without much difficulty. Evolving the field ϕ along the gradient brings a lot of gradient flow methods [36, 49, 65]. Possibly, the most famous one is geodesic curvature flow (GCF) [65], which minimizes the level set length Eq. (1) by the gradient flow:

$$\frac{d\phi}{dt} = -|\nabla\phi|\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{2}$$

Evolving ϕ according to this flow (a particular form of Hamilton-Jacobi equation) contributes a large body of literature.

2.1 Regularization

To prevent the scalar field from being too steep or too flat, people usually enforce $|\nabla\phi| = 1$ by re-initialization [9,10,44], penalty of regularization terms [15, 24, 36, 37], augmented/projection Lagrangian [19, 40] and auxiliary variable [18]. For instance, the following eikonal penalty function is used in [24, 36]:

$$\int_M \frac{1}{2}(|\nabla\phi| - 1)^2 dM. \tag{3}$$

This basically asks that ϕ to be a signed distance field (SDF) w.r.t. the level set $\phi^{-1}(0)$. We adopt a similar strategy in Sec. 3.1.

3 Curve Design by Optimizing Level Set

Different from seeking the shortest curve, the goal of our work is to efficiently design a closed curve without self-intersection with flexible controls via the following optimization problem:

$$E_{\text{total}}(\phi) = E_{\text{shape}}(\phi) + w_{\text{sdf}}E_{\text{sdf}}(\phi), \tag{4}$$

where E_{shape} controls the shape of the curve, accounting for the smoothness, interpolation, tangent *etc*., and E_{sdf} is the regularization term. Below, we list the most important terms and introduce the others along with the experiments. The discretization and derivatives of them can be found in Appendices A and B.

3.1 SDF regularization

To penalize the deviation from SDF, the most common technique is to incorporate the eikonal term Eq. (3) [15, 24, 36]. However, as pointed out in [37], such a formulation has serious **stability issue** because its derivatives become ∞ when $|\nabla\phi|$ approaches 0, which happens frequently when the field is locally flat. Therefore, we use the following modified formulation for regularizing SDF function:

$$E_{\text{sdf}}(\phi) = \int_M \frac{1}{2}(|\nabla\phi|^2 - 1)^2 dM. \tag{5}$$

Taking the square erases the infinity in its derivative as shown in Appendix B.

3.2 Shape control

The shape of the level set can be controlled in different ways by constraining its values or derivatives. Here, we consider some commonly used terms including

$$\begin{aligned} E_{\text{shape}}(\phi) = {} & w_{\text{interp}}E_{\text{interp}}(\phi) + w_{\text{obs}}E_{\text{obs}}(\phi) \\ & + w_{\text{smooth}}E_{\text{smooth}}(\phi) + w_{\text{tan}}E_{\text{tan}}(\phi) + \cdots. \end{aligned} \tag{6}$$

Users can also design their own energy terms regarding their specific requirements.

3.2.1 Interpolation

Enforcing interpolation condition is very simple, which just needs the value of ϕ to be 0 at the specific points $\mathcal{Q} = \{q_i\}$ through a penalty function:

$$E_{\text{interp}}(\phi) = \sum_{i=1}^{|\mathcal{Q}|} \frac{1}{2}\phi(q_i)^2, \quad \forall q_i \in \mathcal{Q}. \tag{7}$$

For the sake of brevity, we use uniform weight for each landmark point in this paper, and assume q_i is always a mesh vertex. Our method can be simply extended to support non-uniform weights and take any point on the mesh as the landmark point q_i represented by their barycentric coordinates.

3.2.2 Obstacle

In addition to interpolation condition, we can also make the curve avoid some regions. This can be achieved by adding a penalty term to the energy function:

$$E_{\text{obs}}(\phi) = \int_\Gamma c\, d\Gamma = \int_M \delta(\phi)c|\nabla\phi|dM \approx \int_M c\delta(\phi)dM. \tag{8}$$

The weight c is set to be a large value near obstacles while 0 for valid regions. We empirically make it 1 in the obstacle center and smoothly decrease it to 0 in a narrow support by a Gaussian kernel. The co-area term $|\nabla\phi|$ is ignored because we strongly penalize the SDF term Eq. (5).

3.2.3 Smoothness

We measure the smoothness by Willmore energy [56]:

$$E(\phi) = \int_\Gamma \frac{1}{2}\kappa^2 dl, \tag{9}$$

where $\kappa = \text{div}(\nabla\phi/|\nabla\phi|)$ is the geodesic curvature of the level set.

The original Willmore energy is too complex to be optimized directly, so we use the approximation:

$$E_{\text{smooth}}(\phi) = \int_\Gamma \frac{1}{2}\kappa^2 dl = \int_M \frac{1}{2}(\Delta\phi)^2\delta(\phi)dM. \tag{10}$$

Because of the SDF term Eq. (5), we approximate geodesic curvature $\kappa(p) \approx \text{div}(\nabla\phi(p)) = \Delta\phi(p)$, i.e. Laplacian of ϕ. The co-area term $|\nabla\phi|$ is ignored here because of the same reason.

3.2.4 Tangent

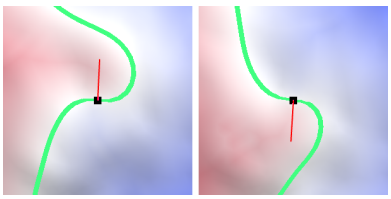
To control the tangent at landmarks, there are two possible ways. One is to make the gradient $\nabla\phi(p_j)$ perpendicular to a specified direction τ_j on the tangent plane $T_{p_j}(M)$ of an interpolation point p_j:

$$E_{\text{tan,t}}(\phi) = \sum_j \frac{1}{2}\left\langle \nabla\phi(p_j), \tau_j \right\rangle^2. \tag{11}$$

Aligning the gradient $\nabla\phi(p_j)$ to a tangent direction $n_j \in T_{p_j}(M), \|n_j\| = 1$ can even specify the exterior side of the curve at point p (when appointing the region with positive ϕ as exterior). In other words, the following term can control the outward normal direction of the curve at p_j:

$$E_{\text{tan,n}}(\phi) = \sum_j \frac{1}{2}\left(\left\langle \nabla\phi(p_j)/|\nabla\phi(p_j)|, n_j \right\rangle - 1\right)^2. \tag{12}$$

As shown in the inset, one can specify the orientation of n_j to choose the left or the right result. Rotating n_j to τ_j in $T_{p_j}(M)$ by ± 90 degree, the term $E_{\text{tan,t}}$ automatically gives the one with lower energy.



This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365

Noticing that $|\nabla\phi| \approx 1$ because of the SDF term Eq. (5), we experimentally approximate Eq. (12) by Eq. (13) to eliminate the numerical complication from the normalization:

$$E_{\text{tan,n}}(\phi) = \sum_j \frac{1}{2}\left(\langle \nabla\phi(p_j), n_j\rangle - C_n\right)^2. \tag{13}$$

If C_n is set to 1 in the above equation, this term can be minimized undesirably by enlarging the gradient $\nabla\phi(p_j)$ (though still close to 1) to compensate the direction misalignment. Setting the constant $C_n > 1$ can alleviate this issue, and $C_n = 2$ works well in all our experiments.

4 Numerical Method

For the unconstrained smooth optimization problem above, we adopt Newton's method.

4.1 Smooth approximation of Dirac Delta

First, the singularity of Dirac function has to be smoothed out for efficient and stable numerical optimization. Thus, the Dirac Delta is usually replaced by smooth approximations [70], here we simply use the Gaussian kernel $G_\sigma(\phi)$:

$$G_\sigma(\phi) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\phi^2}{2\sigma^2}}, \tag{14}$$

where σ is used to control the "support" of smooth Dirac Delta. Obviously, smaller σ gives a more accurate approximation to the "ground truth" problem (i.e. using idea Dirac Delta) at the cost of numerical instability, while larger σ works in an opposite way. The discussion about σ can be found in Sec. 5.1.

4.2 Hessian correction

One major challenge comes from the non-convex nature of the problem, which results in indefinite Hessian. Like [28, 61], we simply project each small Hessian matrix $H_{v_j} = \nabla^2 E^{v_j}$ at mesh vertex v_j into the closest symmetric semi-positive matrix $\tilde{H}_{v_j}$ by clamping negative eigenvalues into zero.

The total Hessian assembled from the Hessian matrices at each vertex may still be ill-conditioned and cause instability when solving the Newton step. We apply Levenberg-Marquardt method [47] to address this issue (see Appendix C for details). Our method yields a nearly quadratic convergence rate in most cases, and a comparison to gradient descent is shown in Fig. 1.

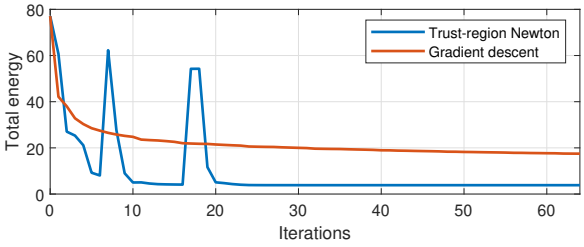


Fig. 1: We compare our numerical method with a simple gradient descent on the same problem, and plot the curves of total energy with iterations for the first 64 steps. Our method yields a much faster convergence rate and much lower total energy.

4.3 Narrow band acceleration

The method described above assembles and solves the linear system over the entire mesh. However, it is only necessary to update the function values in the band region near the current curve, which is usually named as narrow band technique [1,55]. Some methods [41,69] also skip the region with nearly zero gradient.

To construct the narrow band, we first extract faces with opposite ϕ signs at vertices. Then we run BFS to expand the narrow band inside a

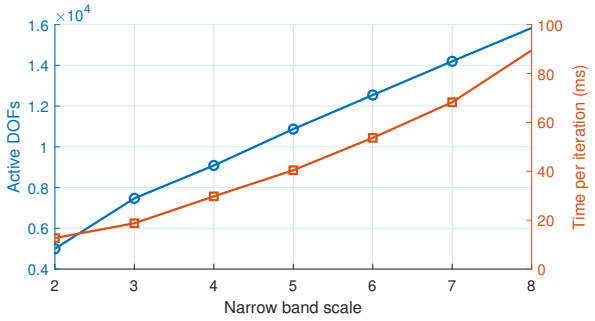


Fig. 2: Statistics about bandwidth. We generate a relatively long curve on a mesh with its length being around $330\bar{l}$. By setting $w_B = k \cdot 3\sigma/\bar{l}, k = 2, 3, \cdots, 8$ and recording the corresponding active DOFs and time per iteration, we can see the linear relationship between w_B and DOFs. Actually, if the length of the curve is $m\bar{l}$, the number of DOFs can be estimated by mw_B. We can also notice that the time cost grows super-linearly with w_B.

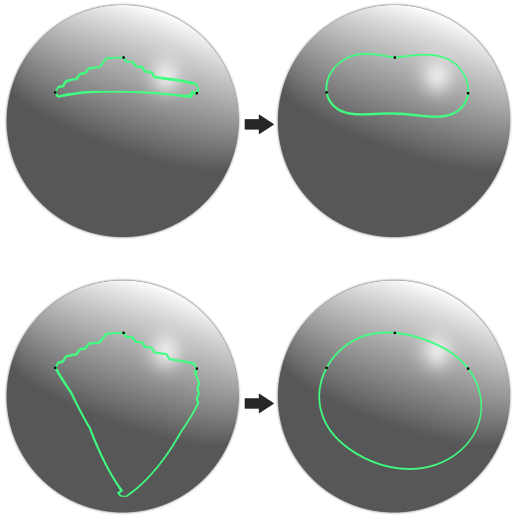


Fig. 3: Our energy function is not convex, so different initial values (left ones) may lead to different results (right ones). Though this behavior is undesired in some scenarios, it helps to keep the interactive curve design procedure more intuitive and controllable.

bandwidth w_B. To achieve better performance, we only update it when there is a significant change in the ϕ values near level set. Specifically, we trigger the update only if both conditions are satisfied:

- for all the faces that the current curve lies on, at least one value on these faces has changed greater than a preset threshold (possibly $\bar{l}$) compared with the previous iteration;
- $|F_k \Delta F_{k-1}| > 0.05|F_k|$, where F_k is the set of faces the current curve lies on, and F_{k-1} is that of the previous round, Δ denoting the symmetric difference between sets.

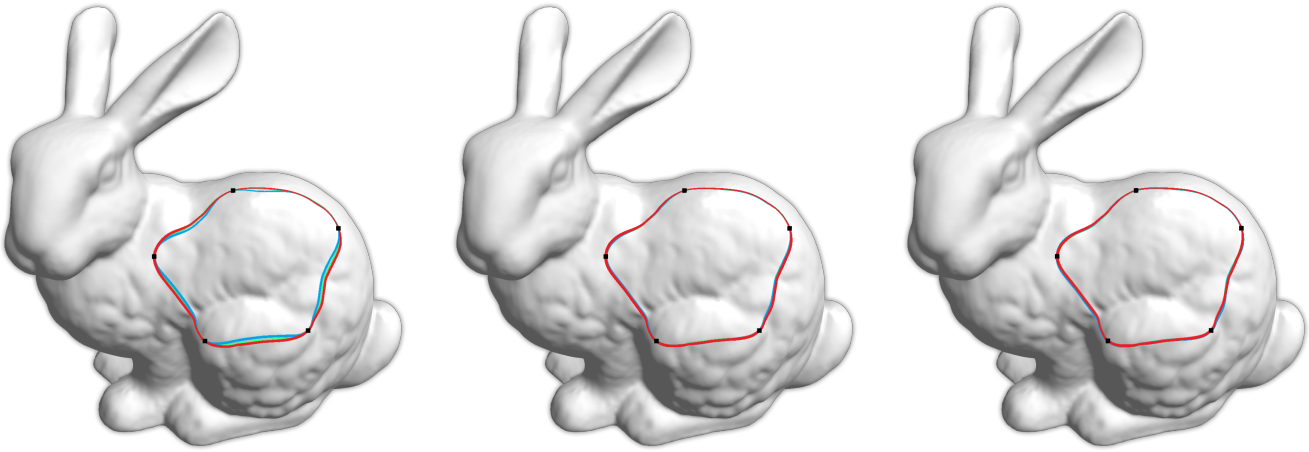
As long as the narrow band is not modified, the cost of Hessian re-factorization is avoided.

Now the number of active DOFs, i.e. the dimension of the sparse linear system, is only the number of vertices in the narrow band. It is easy to see that the DOFs are almost linearly proportional to the bandwidth. Fig. 2 shows some statistics of the algorithms under different narrow band width. In our experiments, we set $w_B = 6\sigma/\bar{l}$ by default.

4.4 Initialization

The initial value of the iterative optimization can be constructed in various ways. For interactive curve editing, after the user picks a sequence

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365



(a) $\sigma = \bar{l}, 2\bar{l}, 3\bar{l}$ (b) $w_{\text{sdf}} = 0.1, 1, 10$ (c) $w_{\text{smooth}} = 10^2, 10^3, 10^4$

Fig. 4: The results achieved by varying one of the parameters ($\sigma, w_{\text{sdf}}, w_{\text{smooth}}$) around the recommended values ($2\bar{l}, 1, 10^3$) and keeping the remaining ones fixed. The three parameter values under each image are for blue, green, red curves respectively. Note that the difference is slight, which means that our method is insensitive to these parameters.

of vertices $q_1, q_2 \cdots q_n$ as interpolation landmarks, we sequentially compute the shortest paths connecting $q_k, q_{(k+1) \bmod n}$ along the mesh edges to form a closed loop without self-intersection. Among this procedure, edge splitting may be required to guarantee self-intersection free. It is also possible to use other mature Bézier-based methods [45, 46] to generate the initial curve. Then, we compute a distance field w.r.t to the initial loop in the narrow band by fast matching, and flip the sign at its one side as the initial value.

It should be noticed that the converged result may depend on the initial value due to its non-convex nature. Like many mesh deformation methods with non-convex energy [29], it may be a desired property in an interactive session, which prevents the shape of the curve from jumping to a very different one (see the supplementary video), but it may also bring trouble in some scenarios like the ones in Fig. 3 though adjusting the support of smooth kernel can alleviate this issue.

5 Results

We implemented the algorithm with C++ and executed it on a PC equipped with i5-13500HX CPU with full parallelization without the help of GPU. The resulting curve (i.e. the level set) is rendered by a fragment shader in GUI. If a polyline in ambient space is required, we trace the zero-contour of ϕ by the method in [30].

5.1 Parameter discussion

There are several parameters in the algorithm. σ in Eq. (14) is empirically set as $2\bar{l}$ in default, where $\bar{l}$ is average edge length of the mesh. σ balances the precision and numerical stability, a possible strategy is to use larger σ during the initial rounds of iterations and then decrease it when close to convergence according to the applications.

The energy weights in Eqs. (4) and (6) balance the influences of different energy terms. For commonly used ones, recommended values are as follows: $w_{\text{sdf}} = 1$, $w_{\text{smooth}} = 10^3$, $w_{\text{interp}} = 10^3$, $w_{\text{tan}} = 1$, $w_{\text{obs}} = 1$. Because the interpolation, obstacle and tangent terms usually serve like hard constraints, we usually fix them as the default value. For other terms, we make an ablation study in Fig. 4. We can see the results are nearly the same given different w_{sdf} and w_{smooth}, which indicates that our method is insensitive to these parameters.

Of course, w_{smooth} does not need to be uniform everywhere. We are able to control the sharpness of the curve by applying non-uniform w_{smooth} at different landmarks, *e.g.*, we can make the curve sharper at the landmark by decreasing the surrounding weights, see Fig. 5.

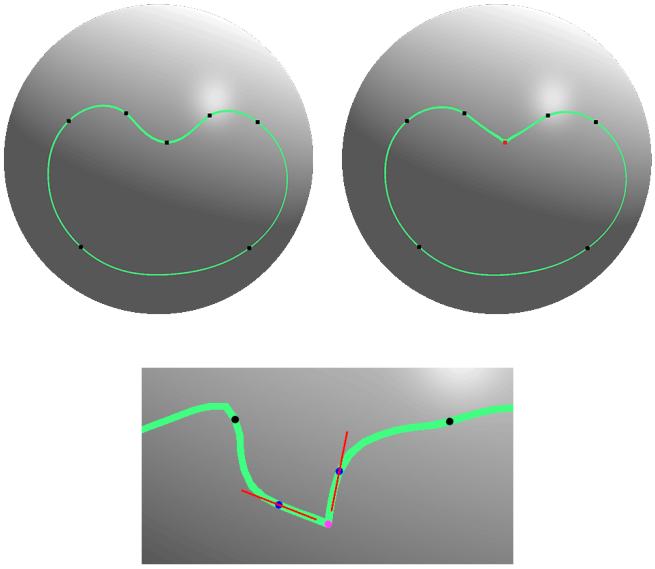


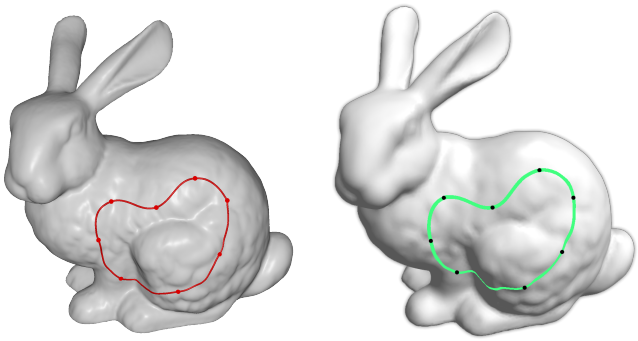
Fig. 5: Applying non-uniform w_{smooth} at different landmarks. The curve on the right is achieved by decreasing w_{smooth} at the red landmark. Tangent control for both sides of the sharp corner can be achieved by simply adding two extra landmarks(blue) near the corner.

5.2 Comparison with spline methods

Spline based curve editing methods [45, 46, 53] are widely used in the surface curve editing. They allow users to edit the curve at interactive rates, while our method does not provide such a real-time performance. However, a quantitative comparison about time and smoothness with them is still hard. Firstly, there is no **optimal** spline interpolation curve, user can always adjust the control tangents to get curve with different smoothness. Secondly, during the construction of splines, user typically add and adjust control points progressively, the pure time cost of algorithm is hard to measure. In Fig. 6, we show that our method can achieve similar results to the Bézier-based method [45] without manual adjustment.

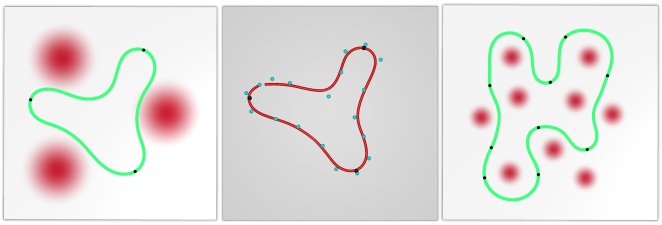
By adding landmarks for interpolation and obstacles, our method can search reasonable smooth results automatically. When it comes to complicated constrained occasions in Fig. 7, the user using spline

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365



(a) (b)

Fig. 6: (a) is a curve generated by the Bézier-based method [45], which uses piecewise Bézier curves to interpolate the landmarks and can be made smooth by empirically aligning the tangents at the seams. (b) we optimize the curve generated in (a) under our energy function. We can see the results are visually similar.



(a) (b) (c)

Fig. 7: (a) shows our result under landmark points (black) and weighted obstacle region (red). (b) is the splines generated by [46] under manually chosen control points (cyan) to "interpolate" the same landmark points in (a). (c) gives our solution under much more constraints, which is cumbersome to be drawn manually by spline methods.

methods needs to carefully design proper control points and tangents to satisfy the constraints, while our method can handle it automatically.

5.3 Comparison to explicit methods

Among variational approaches, we choose [32] here because it shares similar functions with ours, and it is free of projection issue and probably the fastest one due to the assistance of shell surrounding the input mesh.

We ran the two algorithms under the same set of landmarks. And the resulting curves are shown in Fig. 8. In general, the results of our implicit method are visually smoother than those of the explicit method [32]. We further conducted quantitative comparison analysis on the aspects of smoothness and timings.

5.3.1 Geodesic curvature of the resulting curve

Geodesic curvature κ_g is a common measurement of smoothness for curves on manifold. We compute and compare it between the curves generated by the two methods. We first extract the implicit curves into polylines, and then compute discrete geodesic curvature at each vertex using the following formula.

We denote the 3 adjacent vertices on the curve by r_{i-1}, r_i, r_{i+1}, their projections onto the tangent plane at r_i being $\hat{r}_{i-1}, \hat{r}_i, \hat{r}_{i+1}$. Letting θ_i be the angle between $\hat{r}_i - \hat{r}_{i-1}$ and $\hat{r}_{i+1} - \hat{r}_i$, we have

$$\kappa_g = \frac{2\sin\theta_i}{|\hat{r}_{i+1} - \hat{r}_{i-1}|}. \tag{15}$$

We plot the distributions of κ_g in Fig. 8. It can be seen that our implicit method has lower or comparable geodesic curvature along the curve in general since our method better characterizes the geodesic curvature of the curve on mesh instead of simply measuring the Laplacian of each node in a 3D polygon.

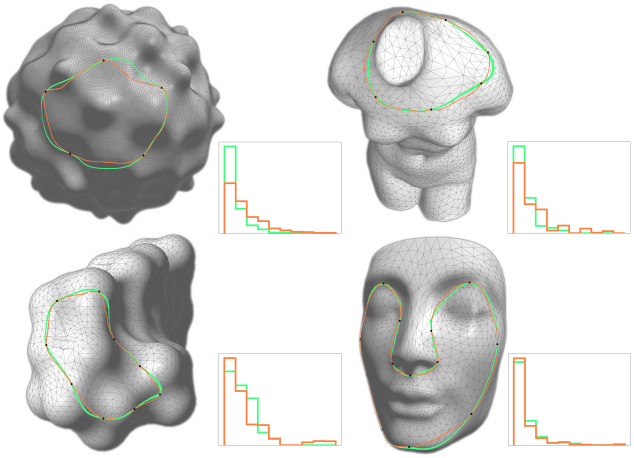


Fig. 8: The curves produced by our implicit method (green), and the explicit method (orange). We first extract the implicit curves into polylines, and then compute the geodesic curvature at each vertex using Eq. (15) for both implicit and explicit curves. The distributions of κ_g are plotted as histograms weighted by segment lengths. Our method generally produces lower or similar geodesic curvature distributions compared to the explicit one.

5.3.2 Timing

During each iteration, both the explicit and the implicit methods require solving a sparse linear system (by Cholesky factorization). In our implicit method, the dimension is the number of vertices in the narrow band ($\sigma = 2\bar{l}, w_B = 12$), while in the explicit method, the dimension of the linear system is 3 times the number of segments on the curve. The sparse matrices in both methods are close to be a banded structure (if the bandwidth is relatively small w.r.t the curve length), so the complexity is around $O(n)$, n being the dimension of the linear system. Here we adjust the proportion of the segment length to the edge length of the mesh to around 0.2, so that the Hessian matrices of the two algorithms have similar dimensions.

Our method is about 7 times slower than the explicit one in each iteration (see the last column in Tab. 1) though we can still achieve over 50 frames per second even for a mesh (bumpy sphere in Fig. 8) with 147K triangles. Please refer to the supplementary video for a demonstration of interactive curve editing.

One major reason of the performance difference is that our sparse linear system has around 17 (average size of two-ring neighborhood of vertices) non-zero entries in each row, but it is only 7 for the explicit method. Furthermore, in the explicit method, the symbolic factorization is only executed once due to the consistent sparsity pattern. However, in the implicit method, re-factorization becomes necessary whenever the narrow band is updated.

The explicit method [32] solves a different optimization problem consisting of the Laplacian and squares of segment length of the 3D polygon. So, it is not surprising that the explicit method converges faster, but the resulting curve has larger maximum curvature, i.e. the curve is less smooth.

Besides the advantage of curve smoothness, our method can be easily applied to an input mesh, while [32] needs to generate a high quality shell space, which is time-consuming and not robust. The other projection-based explicit methods (*e.g.*, [27]) not paying such a pre-computation cost may involve the cost of constructing spatial acceleration structure for projection and face the risk of wrong projection at concave region. In other words, besides the benefit of smoother resulting curve, our method will show much more advantages if the mesh needs to be updated frequently.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365

Model	Method	DOFs	Total cost (ms)	# of Iters	Cost per iter (ms)
Bumpy sphere	Implicit	3139	389	52	7.3
	Explicit	3039	82	67	1.2
Venus	Implicit	668	60	38	1.5
	Explicit	645	3	15	0.2
Retinal	Implicit	780	129	80	1.6
	Explicit	720	13	55	0.2
Nefertiti	Implicit	1245	230	86	2.6
	Explicit	1302	7	16	0.4

Table 1: Comparison with explicit method [32]. The result curves are displayed in Figure 8.

5.4 Applications

Our algorithm can be used in various applications, such as interactive curve editing, curve smoothing, and path planning.

Interactive curve editing We develop a simple graphical user interface to design and edit curves on meshes interactively. An example is shown in Fig. 9. More details can be found in the supplementary video. By virtue of the implicit representation, it is easy to provide Boolean operations of the regions bounded by the curves though we did not provide such an implementation.

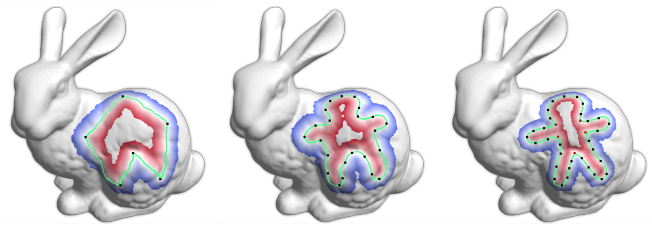


Fig. 9: Drawing a gingerbread man on the bunny. Starting from a rough shape from the left, we refine it by adding landmarks and achieve the desired shape on the right.

Boundary smoothing Many mesh segmentation methods, *e.g.*, [13,48], classify all the faces on a mesh by assigning a class index to each of them. One problem is that the boundary of each patch can only follow the edges of the original mesh, and thus maybe zigzag (Fig. 10 left). One way to deal with it is to extract a nearby geodesic loop [66], but if the boundary is locally convex instead of concave, geodesic loop is obviously not a good choice.

To process the boundary curves with our method, we can convert the class index representation to our initial scalar field using a simple strategy. We first specify a class index i, and for the vertex v_j, we check the faces in its one-ring neighborhood, then

$$\phi_j = \begin{cases} 1, & \text{all the faces are in class } i, \\ -1, & \text{all the faces are not in class } i, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

This is clearly not an SDF, but is enough for our algorithm to work. Then, one can specify a few landmarks to make the curve closely stay on its original position and keep some features. Obviously, the more landmarks are specified, the curve will be less smoothed out. There are plenty of possible ways to specify the landmarks, either manually or automatically (*e.g.*, evenly distributing). Here, we manually select a few for the results shown in Fig. 10.

Path planning In path planning tasks, we sometimes need to confine the curve to only pass through certain areas, which can be easily achieved by adding interpolation and obstacles constraints to our optimization problem. We can see in Fig. 11 that our method can automatically generate a smooth interpolation curve that avoids the obstacles.

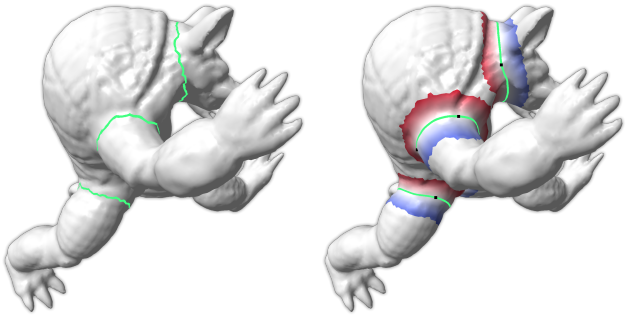


Fig. 10: By adding landmarks and running our algorithm on the zigzag boundaries on the left, we are able to get smoother curves while preserving the main features (right). The right figure also shows how our method works for multiple curves with the narrow band strategy. The segmentation results are from [13].

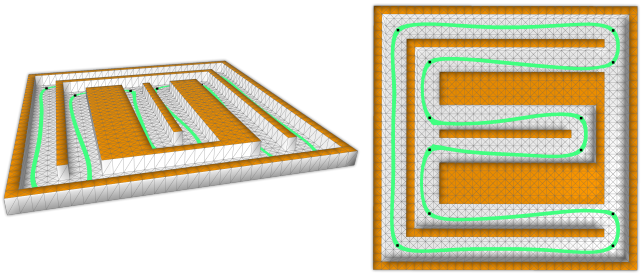


Fig. 11: We generate an E-shaped path under the given landmarks, and we do not want the curve to hit the walls colored in orange.

6 Conclusion

We propose the first implicit curve editing method on triangular mesh with various controls (interpolation, sharpness, tangent and obstacle control). The method can generate curves with smaller geodesic curvature, and has no risk of wrong projection and is efficient.

The major limitation is the lack of precise control over the curve topology. For a given set of interpolation points and the initial value from the expected connectivity among the points, the output of our method may have different connectivity (shown in Fig. 12 middle). To remedy this issue, one can add more interpolation points (Fig. 12 right). Obviously, our method cannot handle intersected curves as well. Finding ways to characterize and control the connectivity could be an interesting future work.

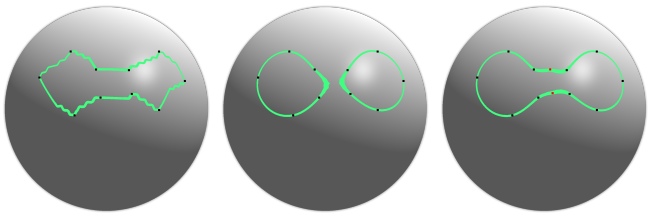


Fig. 12: Given the initial curve on the left, our method will result in the two separate circles in the middle, which is undesired. However, by adding the two red landmarks on the right, we are able to get the expected curve.

Another limitation is about the quality of input mesh. Though our method can handle non-regular and non-uniform meshes (see Fig. 13), meshes with nearly degenerated elements still bring numerical issues, which is similar to most of the algorithms based on field derivatives of a mesh [46].

What's more, the non-convex nature of the energy function will lead

to different results given different initial value. Though this behavior is undesired in some scenarios, it helps to keep the interactive curve design procedure more intuitive and controllable. The time cost of our method is also limited by the non-convexity even if we use Newton's method and narrow band acceleration.

REFERENCES

[1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of computational physics*, 118(2):269–277, 1995. 4

[2] A. Arnould, P.-Y. Gousenbourger, C. Samir, P.-A. Absil, and M. Canis. Fitting smooth paths on Riemannian manifolds: Endometrial surface reconstruction and preoperative MRI-based navigation. In *Geometric Science of Information*, vol. 9389 of *LNCS*, pp. 491–498. Springer, Cham, 2015. 2

[3] L. Bar and G. Sapiro. Generalized newton-type methods for energy formulations in image processing. *SIAM Journal on Imaging Sciences*, 2(2):508–531, 2009. 2

[4] R. Bergmann and P.-Y. Gousenbourger. A variational model for data fitting on manifolds by minimizing the acceleration of a Bézier curve. *Frontiers in Applied Mathematics and Statistics*, 4, 2018. 2

[5] A. I. Bobenko and P. Schröder. Discrete willmore flow. In *ACM SIGGRAPH 2005 Courses*, pp. 5–es. 2005. 1, 2

[6] M. Burger. A framework for the construction of level set methods for shape optimization and reconstruction. *Interfaces and Free boundaries*, 5(3):301–329, 2003. 2

[7] M. Burger. Levenberg–marquardt level set methods for inverse obstacle problems. *Inverse problems*, 20(1):259, 2003. 2

[8] M. Burger and S. J. Osher. A survey on level set methods for inverse problems and optimal design. *European journal of applied mathematics*, 16(2):263–301, 2005. 2

[9] V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische mathematik*, 66(1):1–31, 1993. 3

[10] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997. 3

[11] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001. 2

[12] G. Charpiat, P. Maurel, J.-P. Pons, R. Keriven, and O. Faugeras. Generalized gradients: Priors on minimization flows. *International journal of computer vision*, 73:325–344, 2007. 2

[13] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3), jul 2009. doi: 10.1145/1531326.1531379 7

[14] D. L. Chopp. *Computing minimal surfaces via level set curvature flow*. University of California, Berkeley, 1991. 2

[15] M. Clémot and J. Digne. Neural skeleton: Implicit neural representation away from the surface. 114:368–378, 08 2023. doi: 10.1016/j.cag.2023.06.012 3

[16] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 317–324, 1999. 2

[17] N. P. Dijk, K. Maute, M. Langelaar, and F. Keulen. Level-set methods for structural topology optimization: A review. *Struct. Multidiscip. Optim.*, 48(3):437–472, sep 2013. doi: 10.1007/s00158-013-0912-y 2

[18] J. Duan, Z. Pan, X. Yin, W. Wei, and G. Wang. Some fast projection methods based on chan-vese model for image segmentation. *EURASIP Journal on Image and Video Processing*, 2014(1):1–16, 2014. 3

[19] V. Estellers, D. Zosso, R. Lai, S. Osher, J.-P. Thiran, and X. Bresson. Efficient algorithm for level set method preserving distance function. *IEEE Transactions on Image Processing*, 21(12):4722–4734, 2012. 3

[20] H. Federer. *Geometric measure theory*. Springer, 2014. 2

[21] F. Gibou, R. Fedkiw, and S. Osher. A review of level-set methods and some recent applications. *Journal of Computational Physics*, 353:82–109, 2018. 2

[22] P.-Y. Gousenbourger, E. Massart, and P. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *Jou. of Math. Imaging and Vision*, 61:1–27, 2018. 2

[23] P.-Y. Gousenbourger, C. Samir, and P. Absil. Piecewise-Bézier C^1 interpolation on Riemannian manifolds with application to 2D shape morphing. In *Proc. 22nd Int. Conf. on Pattern Recognition*, pp. 4086–4091, 2014. 2

[24] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pp. 3789–3799. PMLR, 2020. 3

[25] M. Hintermüller and W. Ring. An inexact newton-cg-type active contour approach for the minimization of the mumford-shah functional. *Journal of Mathematical Imaging and Vision*, 20:19–42, 2004. 2

[26] M. Hintermüller and W. Ring. A second order shape optimization approach for image segmentation. *SIAM Journal on Applied Mathematics*, 64(2):442–467, 2004. 2

[27] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics*, pp. 284–293, 2004. 1, 2, 6

[28] J. Huang, T. Jiang, Z. Shi, Y. Tong, H. Bao, and M. Desbrun. ℓ_1-based construction of polycube maps from complex shapes. *ACM Transactions on Graphics*, 33(3):25:1–25:11, June 2014. doi: 10.1145/2602141 4

[29] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics*, 25(3):1126–1134, 2006. 5

[30] A. Jacobson, D. Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 5

[31] Z. Ji, L. Liu, Z. Chen, and G. Wang. Easy mesh cutting. *Computer Graphics Forum*, 25(3):283–291, 2006. 2

[32] Y. Jin, D. Song, T. Wang, J. Huang, Y. Song, and L. He. A shell space constrained approach for curve design on surface meshes. *Computer-Aided Design*, 113:24–34, 2019. 1, 2, 6, 7

[33] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988. 2

[34] K. Lawonn, R. Gasteiger, C. Rössl, and B. Preim. Adaptive and robust curve smoothing on surface meshes. *Computers & graphics*, 40:22–35, 2014. 2

[35] Y. Lee and S. Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3):229–238, 2002. 2

[36] C. Li, C. Xu, C. Gui, and M. D. Fox. Level set evolution without re-initialization: a new variational formulation. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 430–436. IEEE, 2005. 1, 3

[37] C. Li, C. Xu, C. Gui, and M. D. Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE transactions on image processing*, 19(12):3243–3254, 2010. 2, 3

[38] S. Lin, J. Yan, D. Kats, and G. J. Wagner. A volume-conserving balanced-force level set method on unstructured meshes using a control volume finite element formulation. *Journal of Computational Physics*, 380:119–142, 2019. 2

[39] B. Liu, S. Chen, S.-Q. Xin, Y. He, Z. Liu, and J. Zhao. An optimization-driven approach for computing geodesic paths on triangle meshes. *Computer-Aided Design*, 90:105–112, 2017. 1, 2

[40] C. Liu, F. Dong, S. Zhu, D. Kong, and K. Liu. New variational formulations for level set evolution without reinitialization with applications to image segmentation. *Journal of Mathematical Imaging and Vision*, 41:194–209, 2011. 3

[41] Z. Liu, H. Zhang, and C. Wu. On geodesic curvature flow with level set formulation over triangulated surfaces. *Journal of Scientific Computing*, 70(2):631–661, 2017. 1, 4

[42] T. Lolla, P. F. Lermusiaux, M. P. Ueckermann, and P. J. Haley. Time-optimal path planning in dynamic flows using level set equations: theory and schemes. *Ocean Dynamics*, 64:1373–1397, 2014. 2

[43] T. Lolla, M. P. Ueckermann, K. Yiğit, P. J. Haley, and P. F. Lermusiaux. Path planning in time dependent flow fields using level set methods. In *2012 IEEE International Conference on Robotics and Automation*, pp. 166–173. IEEE, 2012. 2

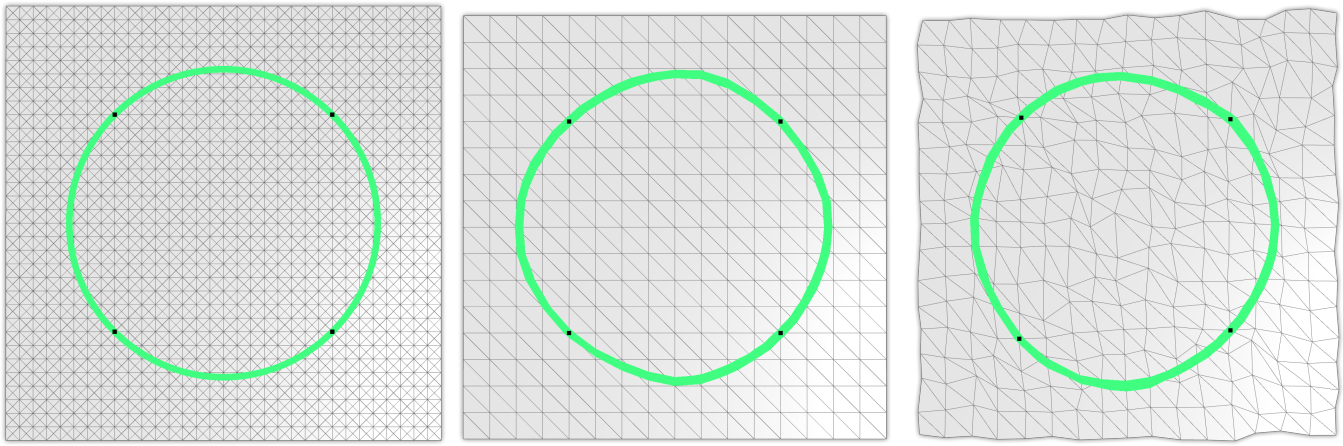
[44] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175, 1995. 2, 3

[45] C. Mancinelli, G. Nazzaro, F. Pellacini, and E. Puppo. b/surf: Interactive bézier splines on surface meshes. *IEEE Transactions on Visualizations and Computer Graphics*, 29(7):3419–3435, 2023. 1, 2, 5, 6

[46] C. Mancinelli and E. Puppo. Computing the Riemannian center of mass on meshes. *Computers Aided Geometric Design*, 103:102203, 2023. 1, 2, 5, 6, 7

[47] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*, pp. 105–116. Springer, 2006. 4, 11

[48] A. Mu, Z. Liu, G. Duan, and J. Tan. Part-to-surface mesh segmentation for mechanical models based on multi-stage clustering. *Computer-Aided*



(a) (b) (c)

Fig. 13: Curves generated on meshes with different resolutions and noise. The distortion is slight, illustrating the robustness of our method.

Design, 162:103545, 2023. doi: 10.1016/j.cad.2023.103545 7

[49] P. Mullen, A. McKenzie, Y. Tong, and M. Desbrun. A variational approach to eulerian geometry processing. *ACM Transactions on Graphics*, 26(3):66–es, jul 2007. doi: 10.1145/1276377.1276459 2, 3

[50] D. B. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 1989. 2

[51] S. Osher, R. Fedkiw, and K. Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004. 2

[52] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations. *J. Comput. Phys.; (United States)*, 79(1), 11 1988. doi: 10.1016/0021-9991(88)90002-2 1, 2

[53] D. Panozzo, I. Baran, O. Diamanti, and O. Sorkine-Hornung. Weighted averages on surfaces. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 1, 2, 5

[54] H. Pottmann and M. Hofer. A variational approach to spline curves on surfaces. *Computer Aided Geometric Design*, 22(7):693–709, 2005. 2

[55] P. Rosenthal, V. Molchanov, and L. Linsen. *A narrow band level set method for surface extraction from unstructured point-based volume data*. Universitätsbibliothek Chemnitz, 2011. 4

[56] M. Rumpf and M. Droske. A level set formulation for willmore flow. *Interfaces and free boundaries*, 6(3):361–378, 2004. 1, 2, 3

[57] M. Slavcheva, M. Baust, and S. Ilic. Variational level set evolution for non-rigid 3d reconstruction from a single depth camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(8):2838–2850, 2020. 2

[58] J. Song, H. Pan, W. Liu, Z. Xu, and Z. Pan. The chan-vese model with elastica and landmark constraints for image segmentation. *IEEE Access*, 9:3508–3516, 2020. 2

[59] G. Sundaramoorthi, A. Yezzi, and A. C. Mennucci. Sobolev active contours. *International Journal of Computer Vision*, 73:345–366, 2007. 2

[60] G. Sundaramoorthi, A. Yezzi, A. C. Mennucci, and G. Sapiro. New possibilities with sobolev active contours. *International journal of computer vision*, 84(2):113–129, 2009. 2

[61] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA '05*, p. 181. Association for Computing Machinery, Los Angeles, California, 2005. doi: 10.1145/1073368.1073394 4

[62] S. Wang and A. Chern. Computing minimal surfaces with differential forms. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021. 2

[63] S. Wang and M. Y. Wang. Radial basis functions and level set method for structural topology optimization. *International journal for numerical methods in engineering*, 65(12):2060–2090, 2006. 2

[64] R. T. Whitaker. A level-set approach to 3d reconstruction from range data. *International journal of computer vision*, 29:203–231, 1998. 2

[65] C. Wu and X. Tai. A level set formulation of geodesic curvature flow on simplicial surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):647–662, 2009. 1, 2, 3

[66] S.-Q. Xin, Y. He, and C.-W. Fu. Efficiently computing exact geodesic loops within finite steps. *IEEE Trans. Vis. Comp. Graphi.*, 18(6):879–889, 2012. 7

[67] R. Xu, Y. Jin, H. Zhang, Y. Zhang, Y. Lai, Z. Zhu, and F. Zhang. A variational approach for feature-aware b-spline curve design on surface meshes. *The Visual Computer*, 2023. 2

[68] N. Yuan, P. Wang, W. Meng, S. Chen, J. Xu, S. Xin, Y. He, and W. Wang. A variational framework for curve shortening in various geometric domains. *IEEE transactions on visualization and computer graphics*, 24(4):1591–1693, 2023. 1, 2

[69] J. Zhang, C. Wu, J. Cai, J. Zheng, and X.-c. Tai. Mesh snapping: Robust interactive mesh cutting using fast geodesic curvature flow. *Computer Graphics Forum*, 29(2):517–526, 2010. doi: 10.1111/j.1467-8659.2009.01621.x 1, 4

[70] K. Zhang, L. Zhang, H. Song, and D. Zhang. Reinitialization-free level set evolution via reaction diffusion. *IEEE Transactions on Image Processing*, 22(1):258–271, 2012. 4

[71] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of computational physics*, 127(1):179–195, 1996. 2

[72] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pp. 194–201. IEEE, 2001. 2

[73] W. Zhu, X.-C. Tai, and T. Chan. Image segmentation using euler's elastica as the regularization. *Journal of scientific computing*, 57:414–438, 2013. 2

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3427365

A Discretization

We discretize the surface by a triangular mesh, and represent the field ϕ piece-wise linearly by a scalar ϕ_i for each mesh vertex v_i. For an integration $\int_M f dM$ over the surface, we approximate it as

$$\int_M f dM \approx \sum_{t_k \in M} |t_k| f(c_k), \tag{17}$$

where c_k is the barycenter of the triangle t_k. For instance, Eq. (5) is approximated into

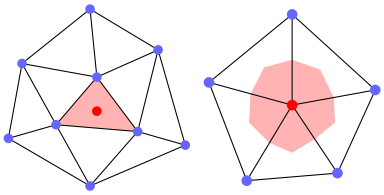
$$E_{\text{sdf}}(\phi) = \sum_{t_k \in M} \frac{|t_k|}{2} (|\nabla \phi(c_k)|^2 - 1)^2. \tag{18}$$

A simple way to discretize Eq. (10) is:

$$E_{\text{smooth}}(\phi) = \sum_{t_k \in M} \frac{|t_k|}{2} (\Delta\phi(c_k))^2 G_\sigma(\phi(c_k)), \tag{19}$$

where the Laplacian at the barycenter is interpolated from the common cotangent form Laplacian $\Delta v_{k,j}$ on three vertices $v_{k,j}$ in triangle t_k, i.e. $\Delta\phi(c_k) = \frac{1}{3}\sum_{j=1}^{3} \Delta v_{k,j}$. But this strategy will lead to relative large stencil for each quadrature point c_k. Thus, we discretize all the integrals on the dual cell of each mesh vertex.

Specifically, the dual cell of a mesh vertex is the star shape that sequentially connects the barycenters of its neighboring triangles through middle of the adjacent edges. Using the face based discretization will bring larger stencil (left). Using the dual-cell based discretization (right), the stencil is just one-ring of a mesh vertex. Under this discretization, the sum of smooth and SDF energies contributed from vertex v_j is:



$$E^{v_j}_{\text{smooth,SDF}}(\phi) = \sum_{t_k \in \mathcal{N}(v_j)} \frac{|t_k|}{6} \Big(w_{\text{smooth}} \Delta\phi(v_j)^2 G_\sigma(\phi(v_j)) + w_{\text{sdf}} (|\nabla\phi(c_k)|^2 - 1)^2 \Big), \tag{20}$$

where $\mathcal{N}(v_j)$ is the one-ring neighborhood of vertex v_j.

B Derivatives

Interpolation term

The derivatives of the interpolation term in Eq. (7) is simple:

$$\begin{aligned} \frac{\partial E_{\text{interp}}}{\partial \phi_j} &= \begin{cases} \phi_j, & v_j \in \mathcal{Q}, \\ 0, & \text{otherwise}, \end{cases} \\ \frac{\partial^2 E_{\text{interp}}}{\partial \phi_j \partial \phi_k} &= \begin{cases} 1, & v_j \in \mathcal{Q} \wedge j = k, \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{21}$$

Smoothness term

The smoothness energy part of Eq. (20) can be rewritten as,

$$E^{v_j}_{\text{smooth}}(\phi) = \frac{|v_j|}{2} \Delta\phi(v_j)^2 G_\sigma(\phi(v_j)), \tag{22}$$

because $\sum_{t_k \in \mathcal{N}(v_j)} |t_k| = 3|v_j|$.

Here we denote the vertices in the one-ring neighborhood of v_j as $u_1, u_2, \cdots, u_m$, and the value of ϕ on them are $\phi_0 = \phi(v_j), \phi_1 = \phi(u_1), \phi_2 = \phi(u_2), \cdots, \phi_m = \phi(u_m)$. The Laplacian at v_j can be written as

$$\Delta\phi(v_j) = \sum_{i=1}^{m} c_i(\phi_i - \phi_0) = \sum_{i=0}^{m} c_i \phi_i. \tag{23}$$

The coefficients $c_0, c_1, \cdots, c_m$ are in widely used cotangent-form:

$$c_i = \frac{\cot\alpha_i + \cot\beta_i}{2}, i = 1, 2, \cdots, m, \text{ and } c_0 = -\sum_{i=1}^{m} c_i, \tag{24}$$

where α_i, β_i are the two angles bounding the edge $\overline{v_j, u_i}$.

Then we have its derivatives as follows

$$\begin{aligned} \frac{\partial E^{v_j}_{\text{smooth}}}{\partial \phi_s} =& |v_j| \Delta\phi c_s G_\sigma(\phi_0) + \frac{|v_j|}{2} (\Delta\phi)^2 G'_\sigma(\phi_0) \delta_{0s}, \\ \frac{\partial^2 E^{v_j}_{\text{smooth}}}{\partial \phi_s \partial \phi_t} =& |v_j| c_s c_t G_\sigma(\phi_0) \\ &+ |v_j| \Delta\phi G'_\sigma(\phi_0)(c_s \delta_{0t} + c_t \delta_{0s}) \\ &+ \frac{|v_j|}{2} (\Delta\phi)^2 G''_\sigma(\phi_0) \delta_{0s} \delta_{0t}. \end{aligned} \tag{25}$$

Tangent term

Following the definition in Fig. 15, we have the expressions

$$\begin{aligned} s =& \sum_{t_k \in \mathcal{N}(p_j)} |t_k|, \\ g_i =& \frac{n_{i2} \times e_{i2} - n_{i1} \times e_{i1}}{2s}, \quad i = 1, 2, \cdots, m, \\ g_0 =& \sum_{i=1}^{m} \frac{n_i \times e_i}{2s}. \end{aligned} \tag{26}$$

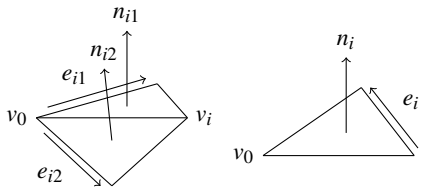


Fig. 15: Some symbols used to define the gradient operator, n_* being the normalized normals of the corresponding triangles and e_* being the edge vectors.

Let $\phi_0, \phi_1, \cdots, \phi_m$ be defined as before. Then we can compute the gradient $\nabla\phi$ in the one-ring neighborhood of p_j by

$$\nabla\phi(p_j) = \sum_{i=0}^{m} g_i \phi_i, \tag{27}$$

and the tangent term and its derivatives have the following expression:

$$\begin{aligned} E^{p_j}_{\text{tan},t} =& \frac{1}{2} \langle \nabla\phi(p_j), \tau_j \rangle^2 = \frac{1}{2} \left(\sum_{i=0}^{m} \langle g_i, \tau_j \rangle \phi_i \right)^2, \\ \frac{\partial E^{p_j}_{\text{tan},t}}{\partial \phi_s} =& \langle g_s, \tau_j \rangle \langle \nabla\phi, \tau_j \rangle, \\ \frac{\partial^2 E^{p_j}_{\text{tan},t}}{\partial \phi_s \partial \phi_t} =& \langle g_s, \tau_j \rangle \langle g_t, \tau_j \rangle. \end{aligned} \tag{28}$$

The derivatives of Eq. (13) are almost the same.

SDF term

We define the gradient operator of t_k by

$$G_k = \frac{1}{2|t_k|} \left[n_k \times e_1, n_k \times e_2, n_k \times e_3 \right], \tag{29}$$

where n_k is the normal of t_k, and e_1, e_2, e_3 are the opposite edge of each vertex. Then the SDF energy in t_k with respect to the values of ϕ at its 3 vertices Φ_k is

$$E_{\text{sdf}}^{t_k} = \frac{|t_k|}{2}(\Phi_k^\top Q_k \Phi_k - 1)^2, \tag{30}$$

where $Q_k = G_k^\top G_k$.

Finally, the derivatives of the SDF energy become:

$$\begin{aligned} \nabla E_{\text{sdf}}^{t_k} =& |t_k|(\Phi_k^\top Q_k \Phi_k - 1) Q_k \Phi_k, \\ \nabla^2 E_{\text{sdf}}^{t_k} =& |t_k| \left(2 Q_k \Phi_k \Phi_k^\top Q_k + (\Phi_k^\top Q_k \Phi_k - 1) Q_k \right). \end{aligned} \tag{31}$$

Even if ϕ is locally flat, the derivatives will not cause infinity.

C Levenberg-Marquardt Method

In Levenberg-Marquardt method, the step $d^{(k)}$ is obtained by solving the linear system:

$$(\hat{H}^{(k)} + \mu_k I) d_N^{(k)} = H_\mu^{(k)} d_N^{(k)} = -g^{(k)}, \tag{32}$$

where I is the identity matrix, and the scalar μ_k is adjusted adaptively following the idea of trust-region method.

To be more specific, we maintain the sequences of the trust-region radius Δ_k and the shift μ_k. At each iteration, a step $d^{(k)}$ is generated using Powell's dog leg method based on the gradient $g^{(k)}$, the Newton step $d_N^{(k)}$, and the radius Δ_k. Then the ratio between the actual energy reduction and the predicted one is evaluated as [47]

$$\rho_k = \frac{E(\phi^{(k+1)}) - E(\phi^{(k)})}{{g^{(k)}}^\top d^{(k)} + {d^{(k)}}^\top H_\mu^{(k)} d^{(k)}/2}, \tag{33}$$

according to which Δ_k and μ_k are updated.