Computers & Graphics (2026)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Extreme continuous level of detail for skeleton-based implicit surfaces

Pierre Hubert-Brierre[a,b,*], Marie-Paule Cani[b], Eric Galin[a]

[a] *Université Lyon 1, CNRS, LIRIS, UMR5205*
[b] *LIX, Ecole Polytechnique/CNRS, Institut Polytechnique de Paris, France*

ARTICLE INFO

Article history:
Received July 13, 2026

Keywords: Implicit surfaces, Level of Detail

ABSTRACT

Skeleton-based implicit surfaces, and convolution surfaces in particular, offer an intuitive and expressive representation for geometric modeling. However, their evaluation and rendering remain computationally demanding, and are often affected by aliasing artifacts. In this paper, we introduce a continuous level of detail framework for the automatic simplification of SCALIS convolution surfaces, achieving up to an order-of-magnitude reduction in rendering time while mitigating aliasing artifacts. Our solution dynamically adapts the blending behavior between shape components to preserve the perceived visual appearance of the surface, while progressively simplifying the underlying skeletal geometry. In contrast to traditional simplification techniques, our method allows controlled changes in topological genus when visually beneficial. Our results demonstrate that the proposed framework effectively reduces visual artifacts and computational cost, while maintaining high fidelity to the overall shape.

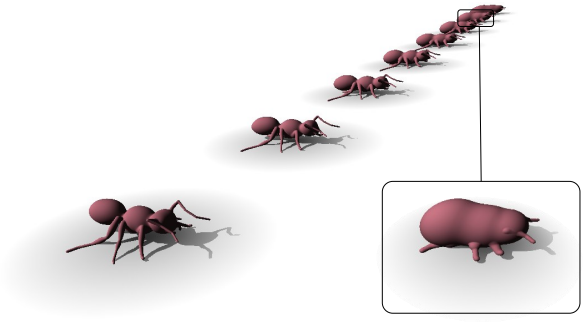

Fig. 1. Given a skeleton-based implicit surfaces (SCALIS model), we automatically generate a continuous set of simplified shapes at different levels of details. Rendering these simplified representations for small screen-space footprints both reduces aliasing artifacts and decreases computation time by more than an order of magnitude.

1. Introduction

Skeleton-based implicit surfaces constitute a powerful representation for shape modeling and editing. They enable smooth blending between primitives defined by an underlying skeletal structure and associated local thickness values. Among other models such as procedurally defined signed distance fields [4] or skeletal-based construction trees [23], advanced convolution surfaces called SCALIS [28] are particularly interesting because of their ability to seamlessly represent both smooth and sharp features. However, evaluating such representations typically requires costly integral computations, leading to limited performance, particularly for complex models.

A variety of strategies have been proposed to accelerate the evaluation of implicit surfaces. Given a field function f defining the surface, these approaches aim at reducing the evaluation cost c_f of field queries $f(\mathbf{p})$ without modifying the resulting implicit surface. Examples include caching redundant computations and optimizing f through spatial data structures such as octrees or proxy geometries [4, 14]. An alternative class of methods relies on level of detail techniques, which explicitly

*Corresponding author:
e-mail: pierre.hubert-brierre@ens-lyon.fr (Pierre Hubert-Brierre)

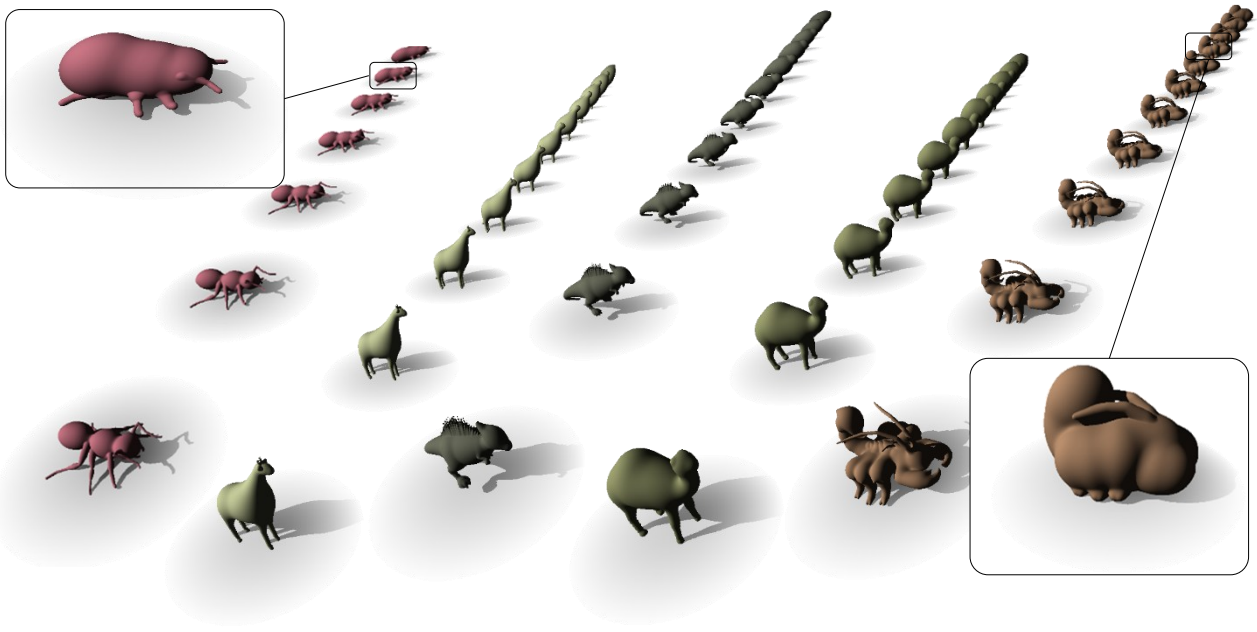

Fig. 2. Our method works on a variety of models, regardless of their topology, allowing for extreme surface simplification and topology changes if necessary.

simplify the underlying shape. These approaches remove or approximate fine-scale features deemed perceptually insignificant, thereby reducing computational complexity and geometric detail. Such filtering-based methods produce smoother representations that are less prone to aliasing artifacts, particularly under distant viewing conditions (see Figure 1 and Figure 2). While level of detail techniques have been extensively investigated for mesh-based representations, they remain largely underexplored for implicit surfaces. In particular, to our knowledge, no prior work enables the automatic construction of continuous levels of detail for convolution surfaces.

In this paper, we present a novel framework for generating continuous levels of detail for the SCALIS model [28]. Our approach supports continuous geometric simplification while allowing topological changes when necessary. This enables a broad spectrum of transformations, ranging from a highly detailed, user-defined surface to a coarse elliptical approximation, while preserving the perceived global shape. In addition to mitigating aliasing artifacts, the progressive removal of unnecessary skeletal primitives significantly reduces the cost of field queries.

The core of our method consists of a two-step transformation applied to the input implicit shape: an inflation phase followed by a deflation phase. The inflation step modifies both the isovalue of interest and the kernel width, effectively smoothing out fine-scale geometric details. The subsequent deflation step restores the overall size of the object. This process is complemented by an automatic pruning of the skeletal structure, further reducing the computational overhead associated with integral-based field evaluations. More precisely, our contributions are as follows: 1) an inflation process designed to generate continuous levels of detail combined with a deflation process that maintains the original size and elongation of the shape; and 2) a dynamic skeleton pruning mechanism, that reduces the model complexity throughout shape simplification.

2. Related work

This section reviews level-of-detail strategies that adapt the geometric complexity of implicit surfaces, and thus computational time, to their size on screen. They include: 1) traditional methods, after conversion of the implicit surface to a mesh, 2) techniques for increasing the level of details when we zoom on an implicit model, 3) approaches for simplifying an implicit model, and gain efficiency for far views.

2.1. Levels-of-Detail through and from meshes

A traditional approach to surface simplification involves converting the surface into a polygonal mesh and subsequently applying established mesh decimation techniques. Luebke *et al.* [18] provide a comprehensive survey of mesh-based methods. While conceptually straightforward and widely adopted, this approach has significant limitations when applied to implicit surfaces. In particular, the loss of the implicit representation makes topological simplification fundamentally more challenging on a mesh than on an implicit surface, while at long distances, blending neighboring parts, such as the legs of a standing character, would perfectly make sense.

An alternative method consists in generating several versions of the mesh on progressively coarser sampling grids when meshing the implicit surface, eventually with sub-meshes joined by linking algorithms such as Transvoxel [17] (see [9] for a comprehensive survey of implicit meshing techniques). However, the result remains a mesh, a static memory-consuming surface representation that can only approximate smooth shapes, therefore inherently reducing the quality of ray-traced images. This is unfortunate, given that direct ray-tracing of implicit

shapes was studied for long [12], was recently accelerated for both skeleton-based [2] and signed distance field models [26], and would leverage the high precision of implicit surfaces. Unlike mesh-based methods, the LoD framework introduced in this work allows for extreme simplification including topological changes while maintaining the implicit nature of the shape. Therefore, all the images in this paper were generated using direct ray-tracing.

One could mitigate these issues by reconstructing an implicit surface from a polygon soup. For example, Kanai employed SLIM surfaces to approximate polygonal geometry using a hierarchy of implicit surfaces with varying levels of detail [16]. In the same spirit, Shen extended the moving least-squares method to incorporate polygonal surfaces approximation constraints, enabling dynamic level-of-detail adjustments via the size of the moving window [20].

In contrast to these works, we address the problem of level of details on implicit surfaces in the context of skeleton-based implicit modeling. Both the input shape and the level of details we generate belong to the same implicit representation, namely SCALIS [28].

2.2. Complementing implicit shapes with details

In implicit modeling, a common level-of-detail strategy involves starting with a rough shape, and then progressively enriching it, by blending smaller geometric features. The different stages of this modeling process can then be stored and reused as intermediate level of detail representations for the final shape. This process however remains manual, except for the few exceptions presented next.

A few works studied the automatic refinement of skeleton-based implicit surfaces, with the aim to make shape parts smoother in close views [8, 1]. Starting from a coarse skeleton graph composed of connected line-segments and/or triangles, subdivision curves or surfaces were used to automatically generate smoother skeletons depending on the zoom factor. However, shape simplification beyond the input skeletal graph was not studied.

Zanni *et al.* [27] addressed the complementary problem of consistently adding sharp pattern-based procedural details to an input, skeleton-based implicit shape.

More recently, Yifan *et al.* [25] introduced a displacement map technique for detail amplification. Global detail amplification was computed by a neural network, significantly improving performance. While effective, theses two methods only provides two distinct representations, the base surface and the detailed shape, and thus do not support any continuous transitions between levels of detail.

Note that enabling continuous transitions between different models is a challenging problem. The simple strategy consisting in using linear interpolation between the initial and detailed scalar fields, such as morphing presented in [7], would require the simultaneous evaluation of the two field, and thus be computationally expensive. More complex interpolation strategies exist for the Blobs [10] and BlobTree models [11], at the expense of more involved computations. In contrast our solution allows for time-continuous transitions between models while reducing the complexity of field queries when the shape is simplified.

2.3. Automatically simplifying an implicit model

To address shape simplification in far views, Barbier *et al.* [3] proposed an automatic method for computing multiple levels of detail while preserving the object topology. However, their approach is integrated within a meshing pipeline and involves substantial per-primitive computations. In contrast, our solution operates directly on skeletal primitives and formulates simplification as a lightweight geometric optimization problem, resulting in reduced computational overhead.

A more general, neural solution was proposed by Takikawa *et al.* [22], who employ a neural network with parameters that vary according to the desired level of detail and the query point. Specifically, they subdivide space using an octree, enabling the network parameters to be adapted locally to each field query. However, neural networks typically require extensive training time and exhibit a substantial memory footprint. In contrast, our method runs in seconds and maintains a minimal memory requirement, thanks to the skeleton-based representation.

In the spirit of displacement mapping, Hubert *et al.* [14] introduced a normal warp operator for signed distance fields created from construction trees to accelerate field queries. Their method leverages a coarse geometry for surface intersection and a detailed one for shading computation. However, the low-resolution models need to be defined beforehand by the user. In contrast, our work addresses the automatic generation of implicit levels of details that could be later be combined with such a mechanism for further accelerating field function queries.

3. Overview

An implicit surface is defined as the c-level set of a continuous scalar field $f : \mathbb{R}^3 \to \mathbb{R}$:

$$S = \{\mathbf{p} \in \mathbb{R}^3 | f(\mathbf{p}) = c\} \tag{1}$$

Implicit surfaces can be categorized into different families. When f is defined by a neural network, the surface is referred to as a neural implicit representation. When f corresponds to a signed distance field relative to the surface, the latter, defined by $c = 0$, is referred to as signed distance fields. In this work, we focus on a third formulation, where f takes its maximal value (often set to 1) at the center of the shape and then decreases to zero with the distance, the surface being defined by some intermediate isovalue such as $c = 0.5$. These surfaces, called blobs [5], meta-balls, or soft-objects [24] when generated by point primitives, where later generalized to skeleton-based convolution surfaces by defining f as the integral of a kernel along complex skeletal structures [6, 28]. This section provides the necessary background, prior to introducing our solution.

3.1. Background: Skeleton based implicit surfaces

Blobs were originally introduced to represent smooth, organic shapes, owing to their inherent ability to blend primitives seamlessly. f was typically constructed as the sum of the fields

f_i, each one being a decreasing kernel function k_i of the distance d to the associated skeletal points S_i:

$$f(\mathbf{p}) = \sum_{i=1}^{N} f_i(\mathbf{p}) = \sum_{i=1}^{N} k_i(d(S_i, \mathbf{p})).$$

This model was soon generalized to the use of arbitrary, n dimensional, geometric primitives for the S_i, such as line segments, arcs of circles, or triangles. In the remainder of the paper we note Θ_i the n_i dimensional parametrization of the skeletal primitive S_i. While they ensure smooth blending between primitives, such *Distance surfaces* however produce undesirable bulging artifacts, due to the summation of fields as two skeletal primitives approach each other. Therefore, they could not be used in association with complex skeletons, such as graphs of connected line segments.

Convolution surfaces were introduced by Bloomenthal *et al.* [6] to mitigate this issue. Rather than defining the f_i as a function of the distance to a skeletal primitive (*i.e.*, distance to the closest point on S_i), the field is computed as the convolution of a kernel k along the skeleton of the primitive, therefore accumulating the field contributions of all skeletal points:

$$f_i(\mathbf{p}) = \int_{\theta \in \Theta_i} k(d(S_i(\theta), \mathbf{p}))\, d^{n_i}\theta.$$

Since the Gaussian kernel k used in [6] made the model inefficient by requiring numerical solving of the convolution integral, subsequent works [19, 21, 8] focused on the search for alternative kernels yielding closed-form solutions. They include:

- The inverse kernel of degree n:

$$k(d) = \frac{1}{\left(\frac{d}{\sigma}\right)^n};$$

- The Cauchy kernel of degree n:

$$k(d) = \frac{1}{\left(1 + \left(\frac{d}{\sigma}\right)^2\right)^{\frac{n}{2}}};$$

- The compact-support polynomial kernel of degree n:

$$k(d) = \begin{cases} \left(1 - \left(\frac{d}{\sigma}\right)^2\right)^{\frac{n}{2}} & \text{if } d < \sigma \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

In these models, the kernel width σ determines the locality of the generated field, while the degree of the kernel controls its sharpness. Note that while the inverse and Cauchy kernels are C^∞, their unbounded support makes them less suitable for modeling local details than the polynomial kernel.

While convolution surfaces eliminate bulge artifacts, they also reduce geometric control, since the target iso-surface is no longer located at the expected distance from the skeleton. To alleviate this issue, subsequent work introduced a user-controlled thickness factor $\tau_i : \Theta_i \to \mathbb{R}^+$ within the convolution integral [15, 13]. However, this formulation still lacks precise control and does not prevent unintuitive behaviors, such as sharp features becoming excessively smoothed-out when blended with larger shape components.

To address this limitation, Zanni *et al.* [28] proposed a new formulation called SCALIS, standing for *Scale Invariant Integral Surfaces*. Based on the intuition of warping the shape to a normalized reference space through a scaling factor $1/\tau_i$, the field was defined by:

$$f_i(\mathbf{p}) = \int_{\theta \in \Theta_i} k\left(\frac{d(S_i(\theta), \mathbf{p})}{\tau_i(\theta)}\right) \frac{d\theta}{\tau_i(\theta)}.$$

The SCALIS model insures scale-invariant details behavior, provides accurate thickness control and achieves consistent blends between large and small features without any smoothing of sharp details. Since such ability to design detailed shapes make good management of levels of detail all the more necessary to limit aliasing, we adopt this representation in this work. In our implementation, we only used segment and point skeletons. We adopted the polynomial kernel in Eq. 2 with $n = 6$ and $\sigma = 2.2$, although our solution could be easily generalized.

3.2. Our approach to level of detail generation

Our goal is to generate visually continuous levels of details on implicit surfaces, enabling to zoom in and out while maintaining their implicit nature, best preserving their shapes, and reducing both aliasing and computational cost in far views.

Let the input SCALIS surface $\mathcal{S}$ be parametrized by the iso-value of interest c (see Eq. 1), the kernel width σ (see Eq. 2) ; the skeleton of the shape $S_k = \{S_i\}_{i=1..N}$; and the thickness functions τ_i, associated to each skeleton part S_i.

To reduce the aliasing in far views, we propose progressively "blobify" $\mathcal{S}$ as the size of the object on screen decreases. This is done by simultaneously increasing the kernel width σ and decreasing the target iso-value c accordingly (see Figure 3, left). Note that this simplification process is continuous and preserves the overall shape, up to a scaling factor. We therefore associate it, with a deflation D of the shape aimed at restoring its initial size on screen.

During this process, fine-scale details are progressively blended into larger shape components, reducing the difference between surfaces generated by complex skeletal primitives and those generated by point primitives. This, in turn, enables progressive skeleton simplification (Figure 3, right): skeletal primitives S_i corresponding to non-salient details at a given level of detail are either reduced to points, merged with neighboring primitives, or removed entirely, thereby decreasing the cost of field evaluation. Indeed, the computational cost depends solely on the number and type of skeletal primitives S_i. In the limiting case where a skeletal primitive S_i degenerates to a point, the associated convolution integral reduces to a single kernel evaluation:

$$f_i(\mathbf{p}) = \frac{1}{\tau_i} k\left(\frac{d(S_i, \mathbf{p})}{\tau_i}\right).$$

In practice, we use a level of detail parameter $\lambda \in [0, 1]$ to control the simplification level, where $\lambda = 1$ refers to the initial shape and $\lambda = 0$ to a fully simplified version. Our approach is to pre-compute the surface parameters $c(\lambda)$, $\sigma(\lambda)$, the skeleton

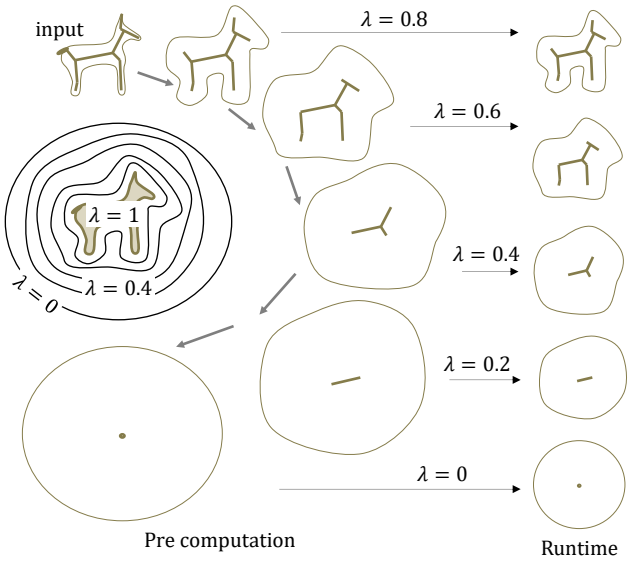

Fig. 3. Overview our method. At the pre-computation stage (left and center), the input implicit shape (top left, $\lambda = 1$) is progressively inflated (circular arrows) by changing both its kernel's width and the iso-value of interest, while the skeletal segments that do not contribute anymore to the displayed iso-value are filtered out to reduce costs. These versions of the shape, continuously parameterized by λ, are rendered in a scaled-down version at runtime (horizontal arrows and right column), as to best maintain the shape's size and aspect ratio. At far distances, a simple ellipsoid may be displayed (bottom right, $\lambda = 0$).

$S_k(\lambda)$, as well as a sampling of the anisotropic deflation parameters $D(\lambda)$. These parameters are then dynamically updated at runtime to create a continuous levels of detail framework.

The approach we just introduced leads to a number of challenges, discussed next: the way the shape and scaling parameters are simultaneously updated to simplify the shape, and therefore reduce its aliasing in distant views, is detailed in Section 4. Conversely, the algorithm used to pre-compute the way the skeleton S_k should be simplified when λ varies, enabling us to reduce the complexity of field queries at run-time, is presented in Section 5.

4. Levels of detail computation

We parametrize the change of levels of detail by a single scalar $\lambda \in [0, 1]$, initialized at $\lambda = 1$. The generation involves a sequential process of inflation and deflation.

4.1. Inflation of the implicit surface

To achieve a harmonious inflation of an implicit shape that progressively smooths and removes fine details, a natural approach consists in modifying the target isovalue. Indeed, smoothing occurs as small high-frequency features are gradually absorbed by the expansion of larger skeletal components. Since field values are maximal near the skeleton and decay to zero outside its region of influence, this process requires decreasing the target isovalue c. At the same time, to prevent the evolving isosurface from reaching the boundary of the primitives' regions of influence, which would inhibit smooth blending during the decrease of c, we correspondingly increase the kernel width (Figure 4).

Our solution keeps these two parameters constant across all skeleton bones. We establish an inverse proportional relationship between the kernel width and the reference iso-value. Specifically, given the user-defined initial targeted iso-value c^* and kernel width σ^*, the inflation parameters c_λ and σ_λ are computed as follows:

$$c_\lambda = \lambda c^* \quad (3) \qquad \sigma_\lambda = \frac{\sigma^*}{\lambda} \quad (4)$$

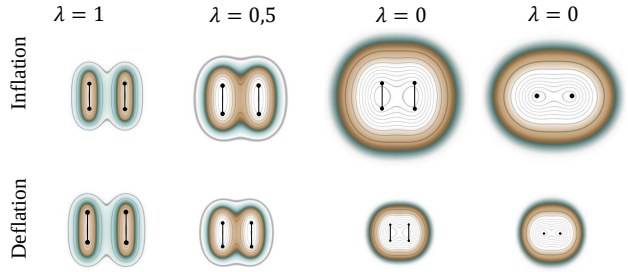

Fig. 4. When the λ decreases (from left to right), we simultaneously lower the targeted iso-surface of interest (the intersection between the blue and brown areas) and increase the kernel width. This inflation can be countered by a deflation step (bottom) and accelerated by simplifying the skeleton when applicable: See the last column, where the line segments collapse into point-skeletons.

It is important to note that, during the inflation (or deflation) step, the thickness τ_i of each primitive S_i remains constant. Surface growth is driven solely by the changes of the iso-value and of the kernel width. Note that while this general approach may look like an arbitrary choice, this solution was found after a number of unsuccessful, unpublished attempts, notably the idea of directly changing the shape thickness without any action on the kernel or on the isovalue. The approach proposed in this work is the only one for which an effective solution was found, although other options will be mentioned in future work.

4.2. Deflation step

In practice, the inflation process both increases the volume of the input shape but also distorts its proportions. To understand the reason for this, let us consider the simple case of the elliptic shape in Figure 5, only distinguishable from other ellipses by its specific proportions and size. Since inflation is direction-agnostic, it does not preserve these proportions. Under uniform inflation by an offset r, the shape's proportion $\alpha_{\lambda=1} = l/L$ evolves as $\alpha_\lambda = (l + 2r)/(L + 2r) \to 1$ as $r \to \infty$.

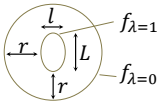

Fig. 5. Inflation affects proportions.

To deflate the shape while preserving its proportions, *i.e.*, by applying an anisotropic scaling $D(\lambda)$, we introduce anchor points defined as the intersections between the surface and level-of-detail axes aligned with the principal axes of the shape (Figure 6). For real-time performance, these anchor points are precomputed on a subset of the level-of-detail surfaces and stored in a lookup table. At runtime, they are interpolated to

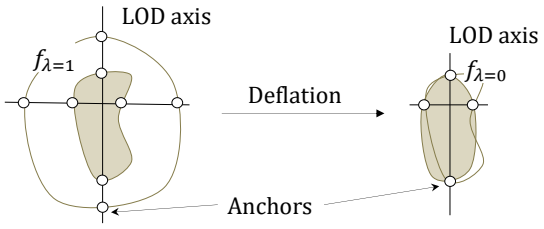

Fig. 6. An anisotropic down-scaling D is used to preserve the positions of the 4 anchor points (respectively, 6 in 3D), thus ensuring that the simplified shape retains the size and proportions of the original one.

derive the anisotropic scaling parameters $D(\lambda)$ required to keep the anchor points fixed in space.

We use an automatic method for computing the best scaling axes, based on the main orthogonal diameters of a sparse sampling of the shape. This involves recursively computing the largest diameter of the shape, projecting the shape onto the orthogonal plane of this diameter, and repeating the process until three orthogonal axes are identified. While this method is both efficient and robust, it may introduce visual artifacts for objects with even proportion, especially when the vertical axis (aligned with gravity) was meaningful (see Figure 7, center). To address these specific cases, users can manually define level of detail axes better suited to the modeled object (Figure 7, right).

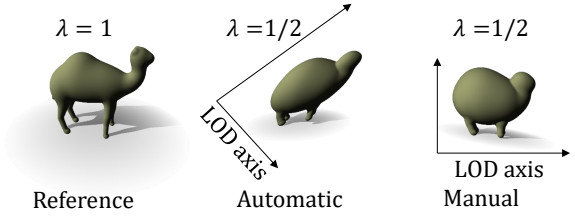

Fig. 7. Impact of the scaling axes.

5. Skeleton simplification

During the surface inflation/deflation process, the sharpest details are progressively smoothed out. As a result, some skeletal primitives may become redundant or contribute only negligibly to the final shape. To optimize computational efficiency, these skeletal primitives are either pruned, merged, or simplified down to a point, thereby reducing the complexity of the smoothed surface representation. To avoid runtime overhead, we precompute a dynamic skeleton $S_k(\lambda)$, whose structure adapts according to the level of detail controller $\lambda \in [0, 1]$.

5.1. Principle and notations

The dynamic skeleton $S_k(\lambda)$ is precomputed by augmenting each skeletal primitive $S_i \in S_k$ with a lifespan interval $\Lambda_i = [\lambda_i^{min}, \lambda_i^{max}]$ and an associated weight w_i (null outside of Λ_i). The lifespan interval Λ_i defines the range of resolutions for which the skeletal primitive S_i is active and should be rendered. The field is then expressed as a weighted sum of the fields $f_i(\lambda, \mathbf{p})$ generated by the active skeletal primitives:

$$f(\lambda, \mathbf{p}) = \sum_{i=1}^{N} w_i(\lambda) f_i(\lambda, \mathbf{p}). \tag{5}$$

A continuous weighting function $w_i(\lambda)$ is used to avoid any sudden change of shape when a skeleton of negligible influence is suppressed, or when a new skeleton primitives appears after a merge.

To facilitate the subsequent discussion, we introduce the following notations. We denote Ω_i the influence region of the skeletal primitive S_i, *i.e.*, the volume where f_i is non-zero. While a negligibility threshold would need to be set if a kernel of infinite radius was used, this region can be easily computed for our choice of the polynomial kernel, as:

$$\Omega_i(\lambda) = \left\{ \mathbf{p} \in \mathbb{R}^3 \mid \exists \theta \in \Theta_i,\ d(S_i(\theta), \mathbf{p}) < \tau_i(\theta)\sigma_\lambda \right\}.$$

For any radius τ, we denote the inflated radius as $\tilde{\tau}(\lambda) = \sigma_\lambda \tau$. Finally, we denote $\bar{S}_i$ the center of the skeletal primitive S_i and $\bar{\tau}_i$ the average radius:

$$\bar{S}_i = \frac{\int_{\theta\in\Theta_i} S_i(\theta) d\theta}{\int_{\theta\in\Theta_i} d\theta} \qquad \bar{\tau}_i = \frac{\int_{\theta\in\Theta_i} \tau_i(\theta) d\theta}{\int_{\theta\in\Theta_i d\theta}}$$

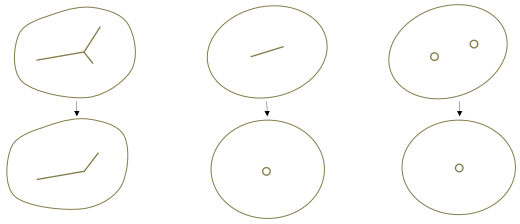

Fig. 8. The three operations used for the simplification of the skeleton are deletion (left column), collapse (center) and merging (right).

5.2. Operations used for skeletal simplification

During dynamic skeletal creation, we make use of the three basic operations, namely, *deletion*, *merging*, and *collapse* (see Figure 8) which are described next. Two of these operations can be tuned through their respective tolerance parameters α_c for collapse and α_m for merging, to balance computational efficiency and visual fidelity according to the needs. In all our experiments, we used $\alpha_c = 1.5$ and $\alpha_m = 1.1$.

Deletion. A skeletal primitive S_i can be deleted at the level of detail $\lambda = \lambda_0$ if there exists another skeletal primitive S_j whose iso-surface at the current iso-value c_λ (defined in Eq. 3) fully includes the bounding volume Ω_i of S_i. Deletion is achieved by setting $\lambda_i^{min} = \lambda_0$.

Collapse. Given a tolerance value α_c, a skeletal primitive S_i, with a maximal radius $\tau_{max} = \max_{\theta\in\Theta_i} \tau_i(\theta)$, can be collapsed at the level of detail parameter λ_0 if its associated iso-surface at the current iso-value c_λ can be bounded by a sphere of radius $\tau_{\tilde{max}}\alpha_c$. In this case, S_i is deleted, and a new skeletal point S_j is inserted in the dynamic skeleton skeletal primitive with radius

$\bar{\tau}_i$, position $\bar{S}_i$ and weight $w_j = \int_{\theta \in \Theta_i} w_i d\theta$. Note that inserting a skeletal primitive S_i at a given λ_0 refers to inserting it within the list of skeletal primitives of $S_k(\lambda)$ with an associated lifespan interval $\Lambda_i = [0, \lambda_0]$.

Merging. Two skeletal points S_{i_1} and S_{i_2} can be merged at the level of detail parameter λ_0 if a hypothetical segment linking the two points can be collapsed with a tolerance of α_m. If so, the two points are deleted and replaced by the skeletal primitive resulting of the collapse of the segment with the weight $w_j = w_{i_1} + w_{i_2}$.

Note that other pruning operators could be thought of, such as collapsing a triangle-skeleton into a segment or merging two segments together. However, we opted for a minimal set of operations, exclusively focusing on those that create skeletal points for their simplicity and fast evaluation.

5.3. Pre-computation of the adaptive skeleton

In practice, the computation of the dynamic skeleton $S_k(\lambda)$ begins by initializing it with the full list of skeletal primitive from the input shape, each assigned to a maximal lifespan interval $\Lambda_i = [0, 1]$ and a unit weight $w_i = 1$.

The sequence of operations to be sequentially applied when λ decreases (respectively increases) is then computed and ordered as follows:

1. Within a first loop, we compute for each primitive and each pair of primitives, the maximal level of detail parameter λ associated with the deletion, merging, and collapse operations, when applicable. All the corresponding pairs (operation, λ value) are stored in the list O.
2. Next, the operation list O is sorted by decreasing λ values and then sequentially applied to update $S_k(\lambda)$. When a skeletal primitive is set to be removed from a given λ_0 and down, all the subsequent operations involving this primitive are suppressed from O. Conversely, when the insertion of a new skeletal primitive is processed at λ_0, the maximal level of detail parameter associated with all the valid operations involving this primitive with other ones are computed and inserted into O.

5.4. Use at runtime

In practice, the control parameter λ associated to each implicit shape in the scene is set from its size on screen (in the companion video, we linearly switched from $\lambda = 1$ when a shape is 400 pixels large or more to $\lambda = 0$ when its size is less than 10 pixels). The pre-computation of the dynamic skeletons ensures that runtime evaluation of each shape $\mathcal{S}$ only processes primitives relevant to the current value λ_0, thereby reducing computational overhead while preserving surface quality. The adequate level of detail to be rendered for $\mathcal{S}$ is generated as follows:

Skeleton. The skeletal primitives S_i that do not contribute to the shape at the current level of details, *i.e.*, such as $\lambda_0 \notin \Lambda_i$, are filtered out from the dynamic skeleton $S_k(\lambda)$.

Weights. If λ_0 approaches either extremity λ_i^* of the lifespan Λ_i (λ_i^{max} or λ_i^{min}), the weight w_i used in Eq. 5 is linearly attenuated within a small interval $\lambda \in [\lambda_i^* - \epsilon, \lambda_i^* + \epsilon]$, such that that $w_i = 1$ within the lifespan and $w_i = 0$ outside of it. Note that if a skeletal primitive S_i collapses into another primitive S_j at a level of detail λ^*, the relationship $\lambda^* = \lambda_i^{min} = \lambda_j^{max}$ holds. Consequently, in the interval $\lambda_0 \in [\lambda^* - \epsilon, \lambda^* + \epsilon]$, the sum of weights satisfies $w_i(\lambda_0) + w_j(\lambda_0) = w_i(\lambda^* + \epsilon) = w_j(\lambda^* - \epsilon)$, meaning the the shapes will fade in and out at the same pace. This conservation property also applies to merge operations.

Inflation and deflation parameters. The shape is rendered using a deflated version of the inflated shape corresponding to the iso-value (Eq. 3) and kernel's width (Eq. 4) at λ_0. In practice, to reduce the computational cost of the deflation stage, we precompute the three downscaling parameters in $D(\lambda)$ along each axis over a sampling of the λ values, store them in a look up table, and use interpolation to get $D(\lambda_0)$ at runtime.

6. Results

We implemented the prunning algorithm in C++. CPU timings were obtained on a desktop com puter equipped with Intel® Core i9, clocked at 3GHz with 32GB of RAM, GPU timings were clocked on an nVidia® GeForce 4070.

6.1. Qualitative results

Figure 9 shows some samples of the levels of detail we generate for six complex SCALIS shapes. Note that, in contrast with the few discrete version often used in static methods, our continuous method may use more than one hundred different skeleton changes during a zoom out sequence, while continuously changing the shape inflation, deflation, and weighing parameters to fully preserve temporal coherence (see the companion video). Changes of topological genus can be observed for the Scorpio and the Dancer models.

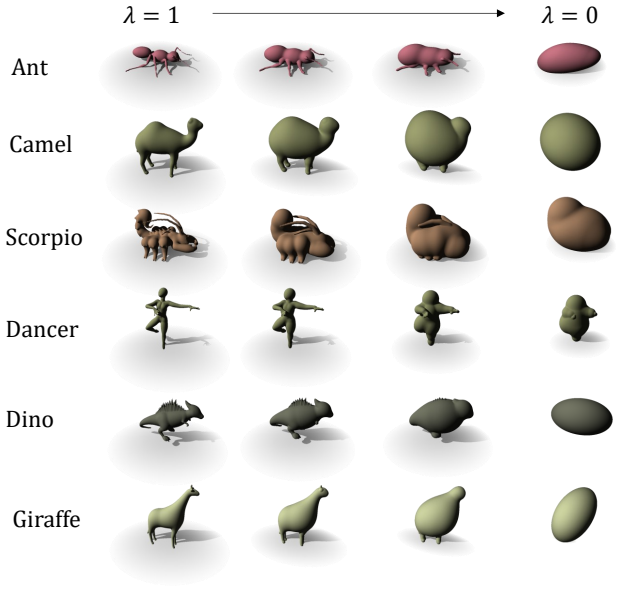

Fig. 9. Continuous levels of details for 6 input shapes (left).

6.2. Performance

Since SCALIS, and convolution surfaces in general, rely on integral computations, the complexity of the skeleton directly impacts the speed of field querie computations. Specifically, simpler primitives such as points are computed faster than segments, which in turn are faster than triangles, even when explicit solutions are available. This explains the performance characteristics of the collapse operation. Overall, by combining all three skeleton update operations, the speedup may reach one to two orders of magnitude in rendering time at low levels of details (see Figure 10).

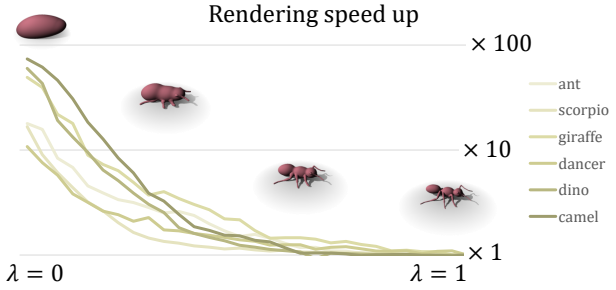

Fig. 10. Speedup in rendering time for each of our models (computed as the time to render the simplified model divided by the time needed to render the baseline). Note the strong acceleration at low resolutions when the surface is smooth enough to be generated by a few skeletal-points.

6.3. Aliasing reduction

In addition to improving performances, simplifying the surface removes high-frequency details, thereby reducing the overall aliasing of the scene. To quantify this reduction, we compared the shape rendered at low resolution with a reference obtained by rendering the same shape at full resolution and subsequently downscaling it to match the low-resolution output. Using root mean squared error (RMSE) as our metric, we observed that our method always reduces the aliasing on our set of examples (see Figure 11).

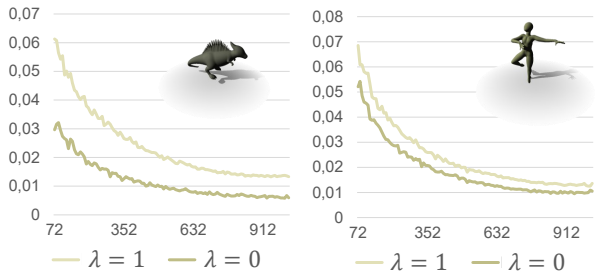

Fig. 11. Aliasing reduction of our method. The curves show the aliasing as a function of screen size in pixel, for specific views of two representative models. Aliasing was measured as the root mean squared error (RMSE) between a high resolution image and a low resolution one, colors being represented as floating value between 0 and 1. Note that as the dancer's fingers are not salient from this viewpoint, the aliasing reduction is lower than in other views. The parameter $\lambda = 1$ corresponds to the baseline, while $\lambda = 0$ corresponds to a simplified surface.

6.4. Limitations

We presented one of the first methods capable of generating a continuous range of levels of detail from SCALIS skeleton-based implicit shapes down to highly simplified representations, typically ellipsoids. Our approach introduces only the geometric and topological modifications necessary to progressively smooth fine-scale details away, while preserving the apparent size and proportions of the shape on screen.

Although the proposed framework provides significant improvements, it also presents several limitations. First, the surface inflation process makes the method unsuitable for acceleration schemes relying on spatial partitioning. Nevertheless, our approach is particularly effective in settings where bounding volume hierarchies are unavailable, such as models defined with unbounded convolution kernels.

Second, the skeleton adaptation algorithm operates by simplifying the underlying field function. Consequently, unlike acceleration techniques that preserve the original field $f(\mathbf{p})$, our method inherently alters the field into $\widetilde{f}(\mathbf{p})$ and deforms the associated implicit surface. However, this trade-off is intentional, as the generation of simplified shapes also contributes to reducing aliasing artifacts.

Finally, the current framework is restricted to rigid shapes. If skeletal primitives were to move relative to one another over time, the precomputed data would become invalid and would need to be extended to account for animation and deformation.

7. Conclusion

In this work, we presented a fully continuous, level-of-detail framework for advanced convolutional implicit surfaces. Our solution involves three precomputed steps: inflation, skeleton simplification and deflation, enabling us to achieve real-time level of detail generation at runtime. Temporal coherence of the displayed shape is achieved in the full range of resolution changes, thanks to the continuous changes of the inflation parameters, the combination of discrete skeletal changes with continuous weighting functions, and an interpolation scheme for the deflation parameters. The robustness of our method enables extreme simplification and topological changes, ranging from fully detailed surfaces to simple ellipsoids. This extreme simplification yields a one to two order-of-magnitude improvement in rendering time. Our method also achieves a strong reduction of aliasing artifacts. Together, these improvements facilitate the efficient rendering of complex shapes, now ready to be combined into crowded scenes.

As discussed in the limitations, our approach is not directly compatible with bounding volume hierarchy optimization. Therefore, future work could explore the development of an inflating strategy for bounding volumes, which could expand in tandem with the surface, potentially addressing this constraint. A second promising direction would be to investigate an optimized variations of various level-of-detail parameters, which could non-linearly depend on the distance to the camera to better preserve the object's shape as the distance from it increases. Another second direction for future work would be an extension to colored or even textured implicit shapes, which

should ideally take both shape and color gradients into account during simplification. Lastly, investigating the computation of a hierarchy of levels of details for animated shapes, and investigating how a skeleton-based animation should be extended or discarded for simplified versions of the skeleton, would be an interesting direction for future work.

Acknowledgments

We would like to thank Karan Singh for early discussions on this problem, and Vincent Bonzac for a first, unpublished solution. We kindly thank Cédric Zanni for sharing his database of SCALIS models with us. This work was partially funded by the project MultIFOrm ANR-22-CE33-0018, supported by Agence Nationale de la Recherche (France).

References

[1] Alexis Angelidis and Marie-Paule Cani. 2002. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications (SMA '02)*. Association for Computing Machinery, New York, NY, USA, 45–52. https://doi.org/10.1145/566282.566292

[2] Melike Aydinlilar and Cedric Zanni. 2021. Fast Ray Tracing of Scale-Invariant Integral Surfaces. *Computer Graphics Forum* 40, 6 (2021), 117–134. https://doi.org/10.1111/cgf.14208 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14208

[3] Aurélien Barbier, Eric Galin, and Samir Akkouche. 2004. Complex Skeletal Implicit Surfaces with Levels of Detail. *Proceedings of WSCG* 12, 1 (2004), 35–42.

[4] Wilhem Barbier, Mathieu Sanchez, Axel Paris, Élie Michel, Thibaud Lambert, Tamy Boubekeur, Mathias Paulin, and Theo Thonat. 2025. Lipschitz Pruning: Hierarchical Simplification of Primitive-Based SDFs. *Computer Graphics Forum* 44, 2 (2025), e70057. https://doi.org/10.1111/cgf.70057 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70057

[5] James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph.* 1, 3 (July 1982), 235–256. https://doi.org/10.1145/357306.357310

[6] Jules Bloomenthal and Ken Shoemake. 1991. Convolution surfaces. *SIGGRAPH Comput. Graph.* 25, 4 (July 1991), 251–256. https://doi.org/10.1145/127719.122757

[7] Jules Bloomenthal and Brian Wyvill (Eds.). 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[8] M.-P. Cani and S. Hornus. 2001. Subdivision-curve primitives: a new solution for interactive implicit modeling. In *Proceedings International Conference on Shape Modeling and Applications*. 82–88. https://doi.org/10.1109/SMA.2001.923378

[9] B. R. de Araújo, Daniel S. Lopes, Pauline Jepp, Joaquim A. Jorge, and Brian Wyvill. 2015. A Survey on Implicit Surface Polygonization. *ACM Comput. Surv.* 47, 4, Article 60 (May 2015), 39 pages. https://doi.org/10.1145/2732197

[10] Eric Galin and Samir Akkouche. 1996. Blob Metamorphosis based on Minkowski Sums. *Computer Graphics Forum (Proceedings of Eurographics)* 15, 3 (1996), 143–152.

[11] Eric Galin, Antoine Leclercq, and Samir Akkouche. 2000. Morphing the BlobTree. *Computer Graphics Forum* 19, 4 (2000), 257–270.

[12] John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

[13] Samuel Hornus, Alexis Angelidis, and Marie-Paule Cani. 2003. Implicit modelling using subdivision-curves. *The Visual Computer* 19, 2-3 (2003), 94–104.

[14] Pierre Hubert-Brierre, Eric Guérin, Adrien Peytavie, and Eric Galin. 2025. Accelerating Signed Distance Functions. *Computer Graphics Forum* 44, 7 (2025), e70258. https://doi.org/10.1111/cgf.70258 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70258

[15] Xiaogang Jin, Chiew-Lan Tai, Jieging Feng, and Qunsheng Peng. 2002. Convolution surfaces for line skeletons with polynomial weight distributions. *J. Graph. Tools* 6, 3 (Sept. 2002), 17–28. https://doi.org/10.1080/10867651.2001.10487542

[16] Takashi Kanai, Yutaka Ohtake, and Kiwamu Kase. 2006. Hierarchical error-driven approximation of implicit surfaces from polygonal meshes. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing (SGP '06)*. Eurographics Association, Goslar, DEU, 21–30.

[17] Eric Lengyel. 2010. Transition Cells for Dynamic Multiresolution Marching Cubes. *Journal of Graphics, GPU, and Game Tools* 15, 2 (2010), 99–122.

[18] David Luebke, Martin Reddy, Jonathan D Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. 2002. *Level of detail for 3D graphics*. Elsevier.

[19] Jon McCormack and Andrei Sherstyuk. 1998. Creating and Rendering Convolution Surfaces. *Computer Graphics Forum* 17, 2 (1998), 113–120. https://doi.org/10.1111/1467-8659.00232 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00232

[20] Chen Shen, James F O'Brien, and Jonathan R Shewchuk. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics* (2004), 896–904.

[21] Andrei Sherstyuk. 1999. Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer* 15, 4 (1999), 171–182.

[22] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11353–11362. https://doi.org/10.1109/CVPR46437.2021.01120

[23] Brian Wyvill, Andrew Guy, and Eric Galin. 1999. Extending the CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum* 18, 2 (1999), 149–158. https://doi.org/10.1111/1467-8659.00365 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00365

[24] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. 1986. Data Structure for Soft Objects. *The Visual Computer* 2 (1986), 227–234.

[25] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. 2021. Geometry-consistent neural shape representation with implicit displacement fields. *arXiv preprint arXiv:2106.05187* (2021).

[26] Cédric Zanni. 2024. Synchronized Tracing of Primitive-based Implicit Volumes. *ACM Trans. Graph.* 44, 1, Article 6 (Nov. 2024), 15 pages. https://doi.org/10.1145/3702227

[27] Cedric Zanni, P. Bares, Ares Lagae, M. Quiblier, and Marie-Paule Cani. 2012. Geometric Details on Skeleton-based Implicit Surfaces. In *Eurographics 2012 - Short Papers*, Carlos Andujar and Enrico Puppo (Eds.). The Eurographics Association. https://doi.org/10.2312/conf/EG2012/short/049-052

[28] C. Zanni, A. Bernhardt, M. Quiblier, and M.-P. Cani. 2013. SCALe-invariant Integral Surfaces. *Computer Graphics Forum* 32, 8 (2013), 219–232. https://doi.org/10.1111/cgf.12199 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12199