Feature-Sized Sampling for Vector Line Art

Stefan Ohrhallinger, Amal Dev Parakkat, Pooran Memari

Pacific Graphics (2023)
R. Chaine, Z. Deng, and M. H. Kim (Guest Editors)

Short Paper

Feature-Sized Sampling for Vector Line Art

S. Ohrhallinger[1], A. D. Parakkat[2] and P. Memari[3]

[1]TU Wien, Austria, [2]LTCI - Telecom Paris, IP Paris, France, [3]CNRS, INRIA, LIX - Ecole Polytechnique, IP Paris, France

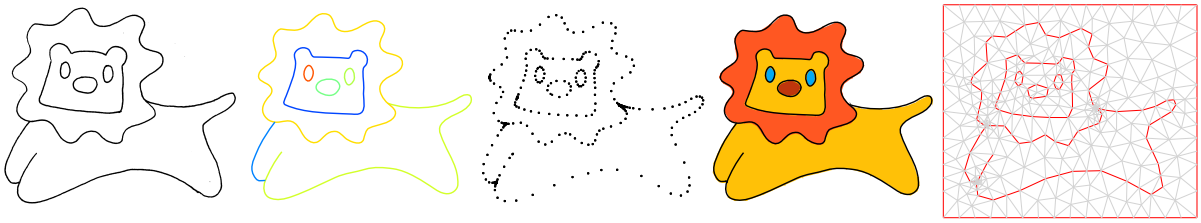

Figure 1: *A scanned sketch, vectorized using [FLB16], ε-samples ($\varepsilon = 0.5$) generated using our method, a result of coloring using (updated) Delaunay Painting algorithm [PMC22], another result after feature-aware meshing with max edge length = 0.1 of bounding box diagonal.*

Abstract

By introducing a first-of-its-kind quantifiable sampling algorithm based on feature size, we present a fresh perspective on the practical aspects of planar curve sampling. Following the footsteps of ε-sampling, which was originally proposed in the context of curve reconstruction to offer provable topological guarantees [ABE98] under quantifiable bounds, we propose an arbitrarily precise ε-sampling algorithm for sampling smooth planar curves (with a prior bound on the minimum feature size of the curve). This paper not only introduces the first such algorithm which provides user-control and quantifiable precision but also highlights the importance of such a sampling process under two key contexts: 1) To conduct a first study comparing theoretical sampling conditions with practical sampling requirements for reconstruction guarantees that can further be used for analysing the upper bounds of ε for various reconstruction algorithms with or without proofs, 2) As a feature-aware sampling of vector line art that can be used for applications such as coloring and meshing.

CCS Concepts

• ***Computing methodologies*** → ***Point-based models; Parametric curve and surface models;***

1. Introduction

Several algorithms have addressed curved reconstruction from unstructured 2D points, sometimes equipped with theoretical guarantees under provided sampling conditions. Usually, these conditions relate to the number of sample points required for a provably correct reconstruction and are not necessarily verified by the given input to the algorithm. The most common sampling condition, called ε-sampling [ABE98], uses the local feature size to capture the features of the curve in a localized manner (details in Section 3).

This work is motivated by a fundamental question in this context: How many points are necessary (as a function of the geometric features of the curve) to represent a curve such that its features can be consistently preserved through a relevant reconstruction procedure? The main practical consequence of such a representation is to design a sampling strategy relating to a generic reconstruction algorithm through a specific sampling condition, i.e. ε-sampling, that ensures a minimum sample points budget while retaining the reconstruction fidelity. Sampling fewer points from a curve reduces processing and storage costs. Additionally, certain applications, such as meshing, require working with discrete points (2D) rather than continuous curves. Therefore, intelligent sampling of the continuous curve while preserving its features becomes essential. Moreover, specific applications, such as sketch coloring, require Delaunay-conforming curves. In such cases, random sampling is insufficient, necessitating intelligent sampling, which constitutes the primary objective of this paper.

This study also serves as a foundation for an analogous surface sampler to better handle common representation issues, such as *redundant* vertices in triangulations, from a sampling perspective.

The main contributions of this work can be summarised as follows:

- An arbitrarily precise ε-sampling algorithm for smooth curves.
- A practical implementation and analysis of the state-of-the-art reconstruction method [BB22] related to ε-sampling guarantees.
- The first evaluation analysis of ε upper bounds for nine existing curve reconstruction algorithms based on 45,000 experiments.
- The first study comparing theoretical sampling conditions with practical sampling requirements for reconstruction guarantees.
- Quantitative evaluation of these curve reconstruction algorithms on a large number of real-world datasets.
- Applications of our sampler for Vector line art coloring and finite element simulations based on Delaunay conforming meshing.

After some preliminaries, we provide an overview of related works, followed by a description of our ε-sampler and a specific implementation in Section 3. Section 4 reports the results of the ε-analysis and the benchmark for nine existing curve reconstruction algorithms. Moreover, possible applications of our proposed sampler in vector line art and finite element meshing simulations are presented in Section 5, before concluding with an outlook on future works. All source code is available at `https://github.com/stefango74/sampling2d`.

Preliminaries

Let us recall some preliminaries in a formal but concise manner. Let C be a set of simple closed smooth planar curves. The medial axis M is the set of all points R^2 having at least two closest points in C [Blu67]. The local feature size $\mathrm{lfs}(p)$ at a point $p \in C$ is the distance from p to its closest point in M of C [Rup93]. A curve C is ε-sampled by a point set S if every point $p \in C$ is closer to a sample $s \in S$ than the ε-fraction of its local feature size $\mathrm{lfs}(p)$, see Fig. 2.

Note that measuring the geometric features of the curve globally, i.e., not just local aspects such as curvature, is classically linked to the computation of the local feature size at curve points, which itself requires determining the medial axis of the curve. To efficiently handle such computation, we will use the fact that many generic curves can be represented or approximated well by piece-wise Bézier curves [Sch07], as shown by BézierSketch [DYH*20]. We recall that Bézier curves of degree n are defined as:

$$B^n(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} p_i, \quad t \in [0,1] \tag{1}$$

By their higher degree, cubic Bézier curves also have higher computational complexity than circular arcs and quadratic Bézier curves, even if fast evaluation algorithms exist [WC20], so their exact medial axis computation is challenging. For our sampling context, we suggest a parameter-based discretization for estimating the local feature size, and a parameter-dependent analysis of its accuracy. This leads to an efficient piece-wise linear approximation of the medial axis for cubic Bézier curves with C^1 tangent continuity (ensuring the smoothness), which offers a user-controllable and quantifiable precision to our ε-sampling procedure.

2. Related Work

We discuss three categories of relevant works here: sampling algorithms for planar curves, works related to the medial axis computation, and curve reconstruction algorithms linked to ε-sampling.

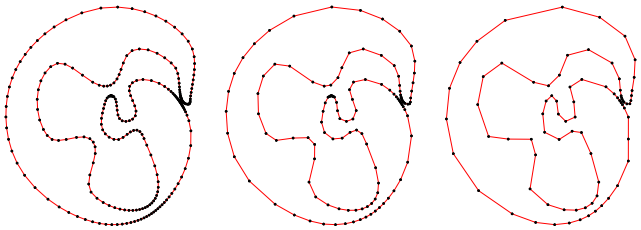

Figure 2: *Example curve sampled with ε=0.33, 0.66, and 1.*

2.1. Sampling of Curves

The sampling of curves has been regarded via different angles in the literature based on error measures, curvature, arc length, critical points, or their combinations - see [PS18] for more details. While distributing points based on absolute error measures (such as RMSE) cannot adapt to small features, local measures like curvatures struggle to capture the global characteristics of the curve.

To the best of our knowledge, the sampling shown in [OMW16] is the only work that considers feature size while sampling. Unfortunately, it suffers from drawbacks such as limited precision (using single floats, which in our experiments show a large error of 10^{-5} in the normal lengths computation), limit on the number of samples (as it has a quadratic runtime complexity w.r.to the sample count) on the discretized medial axis, and lacks an error analysis.

2.2. Medial Axis Computation

The medial axis is, in general, hard to compute since it has higher algebraic complexity than the curve itself, resulting in instability of computation. In other words, small perturbations on the curve may yield large changes in the medial axis. Therefore, algorithms found in the literature usually rely on approximations. Aichholzer et al. [AAA*09] approximate free-form planar curves by a collection of circular arcs in order to quickly compute the exact medial axis of this approximated curve using a divide-and-conquer approach. However, results show high and very localized errors when approximating curves based on Bézier splines. Yang et al. [YBM04] propose an algorithm that also works in 3D and fills the boundary with partially overlapping balls sampling an approximate medial axis and connecting them using corresponding relations on the boundary. Its computational complexity is proportional to the size of this approximate medial axis instead of the boundary, but its accuracy is limited by the number of balls. [RG03] applies boundary marching with user-defined step size to approximate the medial axis instead of tracing bisectors to gain more efficiency in identifying the branching points, while no error analysis is provided. Let us also refer to this survey [ABE09] discussing the challenges and new solutions for medial axis approximation. For our use case, we require complex input as used in real-world examples, such as cubic Bézier curves, and an approximation of the output constructed in reasonable runtime with a quantifiable error.

2.3. Curve Reconstruction Algorithms with Guarantees

In the literature, several curve reconstruction algorithms have been proposed together with theoretical guarantees of how many sam-

ples are required for successfully reconstructing the original curve. These requirements are usually based on the ε-sampling condition, its variants or extensions (in 3D), which gives a quantifiable notion of sampling density as a function of the local feature size. As the next paragraph summarises, many subsequent works have tried to give algorithms that can guarantee correct connectivity under sparser ε-samples (higher ε).

CRUST [ABE98] was the first algorithm with such a proof, requiring $\varepsilon < 0.252$. NNCRUST [DK99] simplified both their algorithm and the proof, enhancing it to $\varepsilon < \frac{1}{3}$. CCRUST [DMR00] also extends CRUST to support open and multiply connected curves. However, their guarantee is more complex as it involves additional parameters. GATHAN [DW01] is the first algorithm that handles sharp corners in practice. Although it comes with a $\varepsilon < 0.5$-sampling condition suggestion, no theoretical proof is provided. As a follow-up work, GATHANG [DW02] adds a distance-based sampling condition to finally prove the reconstruction of sharp corners for $\varepsilon < 0.5$. Later, LENZ [Len06] claims $\varepsilon < 0.48$, but no proof is given for that sampling condition. This long quest continues with the proof of CONNECT2D [OM13] for $\varepsilon < 0.5$ but an additional limit on adjacent edge length ratio of $u < 1.609$, leading to a higher number of requires samples. Then, HNNCRUST [OMW16] manages to prove an enhanced limit of $\varepsilon < 0.47$ while also introducing a new relatable ρ-sampling condition that works better on regions with little curvature, requiring even fewer points at the end. Both CRAWL [PM16] and PEEL [PMM18] only prove generic ε-sampling without a specific limit constant. SIGDT [MOW22] extends the result of CONNECT2D by proving $\varepsilon < 0.5$ with an enhanced $u < 2$. For more details, we refer to the curve reconstruction survey and benchmark [OPP*21] where many of these algorithms are evaluated w.r.t. real-world, and ε-sampled curves. In most reconstruction scenarios, one can globally notice the lack of upper bound analysis (naturally challenging) for ε. The recently proposed algorithm COMPATIBLECRUST [BB22] proves the currently best limit of $\varepsilon < 0.66$, together with a counterexample of $\varepsilon = 0.72$ that is shown not to be reconstructible unambiguously, and therefore limiting the upper bound of ε-sampling.

Despite all the existing theoretical proofs, no practical sampling strategy was available before this current work to precisely evaluate the proposed reconstruction methods under quantifiable sampling conditions. Thanks to the sampling control provided by our ε-sampler, we provide for the first time an evaluation of the tolerated sampling sparsity of the state-of-the-art curve reconstruction methods in the context of ε-sampling. This is done through a detailed analysis of their corresponding upper bounds for ε, see Section 4.1.

3. Our method: ε-Curve-Sampler (ECS)

We propose a quantifiable ε-sampling framework in three steps.

1. Medial axis approximation by polylines
2. Local feature size estimation
3. ε-samples generation

We approximate the medial axis by a set of polylines using a discrete walkaround along the curve, leading to a parameter-based estimation of the local feature size. To progressively generate samples, one generic strategy is to grow feature-sized disks along the curve touching the previously generated sample as long as it fulfils the ε-sampling condition and defines the other intersection as the next sample. An efficient implementation is provided for the specific case of approximation by cubic Bézier curves. We elaborate later on how it can be adapted to curves that are not C^1 continuous.

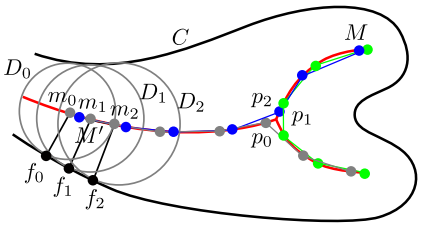

Figure 3: *Stepping along curve C at foot points f_i to determine corresponding points m_i on the medial axis M (red), which are connected along the walk to form a set of differently colored* medial polylines p_0, p_1, p_2 *as a piece-wise linear approximation M' of M.*

3.1. Medial Axis Approximation by Polylines

Given a smooth curve C, the medial axis branches out into its features, forming a graph that would require a tree-like representation. However, we observe that in order to approximate the local feature size of C, branching vertices of the medial axis do not need to be handled globally. In other words, given a prior precision parameter, a local one-sided polyline approximation of (the branches of) the medial axis is sufficient to estimate the local feature size (lfs). To ensure this, we assume that the default stepping parameter is small enough with respect to the finest feature of the curve (the so-called *reach* [Fed59]). Then none of the features of the curve are missed while stepping and generating foot points along the curve, knowing that the exact local feature size computation in a discrete setting is an ill-posed problem by definition. Based on that observation, we suggest the following procedure to efficiently compute the approximated *medial polylines*:

We step along the given curve C in parametric intervals to determine foot points f_i. For each foot point f_i, the corresponding point m_i on the medial axis M is then computed as the center of the largest disk D_i that is tangent to C at f_i and empty of points in C. Implementation details on how this is computed by intersecting with each curve segment are given in the supplement, Section 7.1. Successive m_i are connected as long as they remain on the same side of the curve since a change in the sign of the curvature will place the center of the disk on the opposite side of the curve. Since the generation of medial axis points follows the curve segments, each medial axis segment will be sampled twice (from both sides of the curve as we walk around it). As mentioned before, in the context of local feature size estimation, there is no need to merge such one-sided polylines nor to handle their branching connectivity. We only store the order of the generated points and note their orientation (inside or outside of the curve). In this efficient encoding, two neighbor medial points are connected unless their orientation differs. This forms the linear piece-wise set of polylines M' that approximates M (Figure 3). In Section 3.4, we show an analysis of this approximation error that, by definition, depends on the step

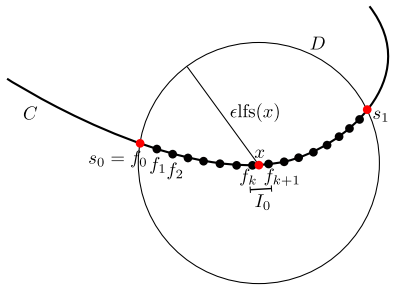

Figure 4: *Sampling: Starting from s_0, f_k is the farthest foot point that fulfills the ε-sampling condition, $|s_0, f_k| < \varepsilon lfs(f_k)$. We then bisect the interval $I_0 = [f_k, f_{k+1}]$ to compute the more precise farthest $x \in C$ with $|s_0, x| < \varepsilon lfs(x)$, and then determine the next sample point s_1 as the second intersection of the disk $D(x, \varepsilon lfs(x)) \cap C$.*

size between foot points, which is assumed small enough with respect to the minimum lfs of the points on the curve (the reach). This parameterization-based precision control is practically convenient to preserve the features of the cubic Bézier curves (which have at most two inflection points).

3.2. Local Feature Size Estimation

For any point $p_i \subset C$, the local feature size $\text{lfs}(p_i)$, being defined as the distance of p_i to the medial axis M, can be instead estimated as the distance of p_i to the set of medial polylines M'. For this, let $V(M')$ denote the set of vertices of M', n_i be the nearest neighbor of p_i in $V(M')$. We can then reduce the computation of the closest point of p_i to M' to the incident edges to n_i in M'. There are at most two of those, based on our polyline-based representation of M'. The distance of p_i to M' is then simply obtained by a projection of p_i onto those edges and computing the minimum distance to p_i.

3.3. Generating ε-Samples on a Smooth Curve

We aim to distribute samples on the curve C so that the ε-sampling condition is fulfilled everywhere and the geodesic distance between all pairs of consecutive samples is maximized to generate the fewest samples possible. Therefore, starting with a sample $s_0 \in C$, we want to determine the farthest point $s_1 \in C$ along a consistent direction such that for any point x_i on the curve segment $]s_0, s_1[$, the ε-sampling condition is fulfilled.

Since we cannot determine s_1 with a closed-form computation based on s_0, we step along C in parametric intervals at the subsequent foot points f_j (reused from the medial axis computation) starting from $f_i = s_0$ to test the sampling condition. To achieve this, first, we want to find the f_k which is geodesically farthest from s_0 and still fulfils the $|f_k, s_0| < \varepsilon \text{lfs}(f_k)$ condition. $\frac{|x,s_0|}{\varepsilon \text{lfs}(x)}$ is monotonic at least for some time (for $|x - s_0| < \delta$ for some $\delta > 0$). This is why you need a fine enough stepping time. This stepping parameter n was chosen carefully to approximate well within the numerical precision of the double float range, see Table 1. Since foot points are only pre-computed at discrete intervals, we further bisect the curve interval $I_0 = [f_k, f_{k+1}]$ to locate the point $x \in I_0$ with maximum $|x, s_0| < \varepsilon \text{lfs}(x)$ iteratively, up to a specified precision (Fig. 4). The next sample s_1 is then located at the second intersection of the disk $D(x, \varepsilon \text{lfs}(x))$ with the curve C.

count	time	MA_σ	lfs_σ	$dist_\sigma$
[OMW16] 10	0.004	0.018	0.0459	0.706
[OMW16] 100	0.051	0.008	0.0429	0.491
[OMW16] 1000	4.786	0.112	0.1208	0.482
[OMW16] 10k	477.237	0.272	1.998	0.612
Ours 10	0.016	1.38e-5	0.0026	0.212
Ours 100	0.235	2.55e-5	0.0006	0.121
Ours 1000	3.329	1.46e-6	0.0005	0.097
Ours 10k	44.405	-	-	-

Table 1: *Sample count per Bézier curve segment and total runtime (in seconds) for* COMPLEX *curve, and comparing with Ours 10k samples for ours and inaccurate algorithm [OMW16]: RMSE of medial axis M, local feature size, and distance between subsequent samples as a function of the longest curve segment hull diagonal.*

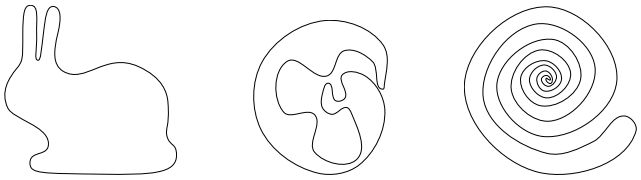

Figure 5: *Example curves consisting of cubic Bézier segments:* BUNNY, COMPLEX *and* SPIRAL.

3.4. Error and Runtime of ε-Sampling Cubic Bézier Curves

In this section, we analyze the runtime and precision of our proposed sampler, including medial axis and local feature size approximation, to highlight the tradeoff between precision and efficiency and show our advantage over the early sampler prototype [OMW16] that has also been used in this benchmark for the generation of test data [OPP*21].

Medial axis: We sample the medial axis with the desired sampling density and connect the samples as sets of polylines. Then, we measure the distance of the ground truth samples to that curve by locating the closest vertex of the curve with a nearest-neighbor search algorithm. Finally, we determine whether their incident edges contain a point that is closer than that vertex. The computational complexity is $O(kn \log \frac{n}{k})$ for k curve segments and n foot points in our implementation. Note that in our experiments, the value of k is relatively small. We additionally reduce the run-time several times by bounding the curve segments with their convex hull (see hull test in Table 2). However, for larger curves, the complexity can easily be reduced to $O(n \log n)$ by using a kd-tree for the bounded segments.

Local feature size and sampling: We compare the local feature size directly at the sampled foot points that are contained in both sample sets, with a computational complexity $O(n \log n)$, for n foot points, since we use a kd-tree for nearest neighbor lookup. However, the computational complexity of sample generation is $O(n)$.

Runtimes: Overall, the runtime is bounded by the medial

Curve	segments	runtime	no hull test	with Eigen
BUNNY	68	0.64	2.22	4.52
COMPLEX	108	4.36	10.76	26.20
SPIRAL	152	3.00	13.91	17.86

Table 2: *Run times in seconds for several curves, with their Bézier curve segment count, without bounding hulls and using Eigen lib.*

axis/lfs computation cost, which is $O(n \log n)$. We also show runtimes for some curve examples (see Figure 5) in Table 2 and the advantage of the specialized quintic solver over the general Eigen library [eig], as well as some results without curve bounding hulls.

Precision setting Since it is infeasible to compute the exact medial axis and, subsequently, the local feature size, we will evaluate their precision using an extremely dense sampling of 10,000 samples per Bézier curve segment as ground truth. This permits comparing with sampling densities that can be run in a reasonable time for the brute-force search and shows their precision/runtime trade-off (Table 1). The analysis shows that the precision increases significantly from 10 to 100 but less so on to 1000 samples per segment. Thus, we used 1000 samples per curve segment in our setting ($n = 1000k$). Comparing with the prototype sampler [OMW16], one can see that we increased the precision of the lfs computation by orders of magnitude while reducing the time complexity from quadratic to logarithmic, subsequently enabling a much more precise ε-sampling of the curve, with a clear quality/runtime tradeoff.

4. Reconstruction guarantees: from theory to practice

The provided feature-aware quantifiable sampling algorithm allows us to analyze for the first time, the performance of the existing reconstruction methods, under supervised ε-sampling settings, where the order of the samples on the original curve is recovered accurately as in the ground truth. This is a remarkable contribution in the context of curve reconstruction to evaluate the quality of the known theoretical bounds on ε and to compare the practical performance of existing methods in terms of guarantees under supervised and quantifiable sampling conditions on input. As we will see, this evaluation confirms that the proven guarantees are often too weak compared to what can be reconstructed in practice. The best theoretical bound for ε was recently provided in the theory-only algorithm COMPATIBLECRUST [BB22] for $\varepsilon < 0.66$, together with a limit example of $\varepsilon = 0.72$ for which no unambiguous curve can be reconstructed. In order to have a first ε-sampling-based evaluation of the state-of-the-art methods, we have implemented COMPATIBLECRUST to complete the whole range of existing codes containing nine reconstruction algorithms.

Our evaluation settings sample the space of $\varepsilon \in [0.5, 1]$ to generate a rich set of input point configurations for our experiments. Using these inputs with various quantifiable sampling properties, we perform an evaluation study that allows us to limit the upper bound of ε, closely approaching the theoretical maximum of 0.72.

We also observe how well reconstruction works in practice even if the guarantee is not fulfilled. Finally, we show how the theoretical state-of-the-art algorithm COMPATIBLECRUST performs in practice on challenging real-world datasets, using the established curve reconstruction benchmark [OPP*21] to determine the percentage of the correctly reconstructed curves (Figure 6). All experiments were run on an AMD Ryzen 7 5800X 8-Core with 4.1 GHz.

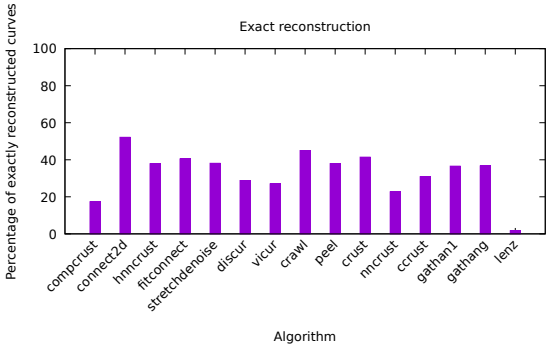

Figure 6: *Benchmark with 2000+ real-world curves [OPP*21].*

4.1. Analysis: Upper Bounds of Algorithms for ε-Sampling

We sample a single curve representative of various features (with differing scales, also opposing each other, and nested, see Figure 5, center) with ε-sampling of $\varepsilon \in [0.5, 1]$ of step size 0.01, and 1000 steps per cubic Bézier curve segment. In order to augment the number of point configurations, we also change the start of the sampling of that curve to varying offsets in steps of 0.01 of the parametric length of the entire curve. This results in 5,000 sampled point sets that cover a very wide range of point configurations through the density and start point variations, equivalent to testing many individual point sets (an entirely different set from the real-world data sets evaluated in Figure 6). Therefore, in total we run 45,000 experiments for the following reconstruction methods: CONSERVATIVECRUST, COMPATIBLECRUST, CONNECT2D, CRAWL, CRUST, HNN-CRUST, NN-CRUST, PEEL, and SIGDT.

Algorithm	lower ε	upper ε	top ε	valid
CONSERVATIVECRUST	-	< 0.5	0.90	27%
COMPATIBLECRUST	**0.66**	**0.74**	0.92	**62%**
CONNECT2D	-	< 0.5	< 0.5	0%
CRAWL	-	< 0.5	0.51	0%
CRUST	0.25	0.60	0.85	49%
HNN-CRUST	0.47	< 0.5	**0.95**	12%
NN-CRUST	0.33	< 0.5	0.71	47%
PEEL	-	0.66	0.79	41%
SIGDT	-	< 0.5	0.73	4%

Table 3: *Analysis: known lower bounds for ε (existing proofs), practical upper bounds for ε (based on our experiments) for correct reconstruction under ε-sampling setting, the top observed ε for any correct reconstruction, and the percentage of such correct outputs integrated overall ε-samples per algorithm in* $\varepsilon \in [0.5, 1]$.

Thanks to our quantifiable sampler, this study constitutes a brute-force test for failure cases of existing reconstruction methods for which only some weak lower epsilon bounds were proved in the past and caps their ε upper bounds in a practical manner, Table 3. It shows that while COMPATIBLECRUST has the highest theoretical

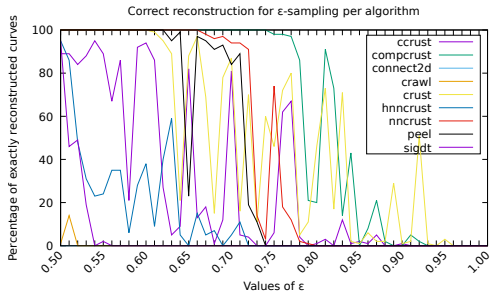

Figure 7: *Practical correct reconstruction of ε-sampled curves.*

guarantee, it performs worse for real-world examples, where CONNECT2D and SIGDT perform best, although having low bounds. CRUST and PEEL perform quite well in both theory and practice.

Figure 7 shows a chart with the detailed results of the brute-force test as the percentage of success as a function of ε, for a set of algorithms that we expect to relate to ε-sampling (not all from the benchmark [OPP*21]). Several sample configurations may still achieve perfect reconstruction even when the epsilon value exceeds the theoretical bound. Therefore, the resulting graph is noisy. The runtime for generating a single sample (using $\varepsilon = 0.5$) of the COMPLEX test set is $\approx$ 4s, resulting in a total runtime of $\approx$ 50h for this analysis study.

5. Applications

This section presents two applications of our sampling framework: fast approximate meshing with Delaunay conforming guarantees for vector line art coloring and finite element simulations (most Figures are located in the supplement).

5.1. Coloring Vector Line Art

Coloring line art is a challenging and time-consuming task, particularly when the line art contains gaps. This challenge is further worsened when working with vector line art, where having gaps is common. Additionally, coloring vector art demands high accuracy (especially for detailed and complex art) and requires precise boundary selection to get the desired result. Even though vector line art is extensively used in the 2D animation industry, existing coloring methods [SDC09, PMC22] are particularly designed for raster sketches. This means the vector art must be rasterised first to employ these methods, particularly since there is no such easy-to-color alternative available for vector line art.

However, converting a vector sketch to a rasterized version for coloring is not ideal since it results in a loss of scalability and quality. Also, it requires a high resolution to capture small features, which in turn requires processing a huge number of pixels. To overcome these limitations, we extend the Delaunay-Painting approach [PMC22] - which is demonstrated to work well on raster sketches with gaps - to handle vector line art. As explained in [PMC22], the core idea remains the same: the user-specified color hint is intelligently propagated through the Delaunay triangles generated from the input pixels to give the desired coloring.

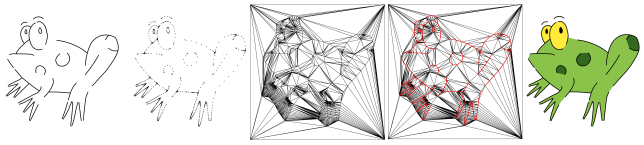

Figure 8: *Left to right: A vector sketch, Samples generated using our method ($\varepsilon = 0.5$), Delaunay triangulation with padded borders, Delaunay superimposed with strokes (demonstrating the Delaunay conforming property), Final colorization.*

Unlike raster contours, where each pixel corresponds to a point in the Delaunay triangulation, sampling rules are not defined for vector line art. Although works like TriWild [HSG*19] can be employed to sample line arts, there is no guarantee that these samples can be used for Delaunay Painting as the Delaunay edges might not consistently align with the vector strokes. Thanks to our sampling method, which leverages the advantageous Delaunay conforming property of ε-sampling, our output can be effectively used as an input to Delaunay Painting (Fig. 8). Moreover, it stays consistent even when the value of ε is changed (Fig. 11). To prevent color bleeding across the boundary (which cannot be quantified just by the edge length as in [PMC22]), we enforce an additional constraint that the color should not spread through edges shared by adjacent samples in the same curve. Figure 14 shows a few results generated on vector line art (original inputs were taken and vectorized from [PMC22]) using our updated version of Delaunay Painting.

5.2. Fast Approximative Meshing

As another example, we show the effectiveness of using our sampling in meshing 2D curves, permitting interactive simulations with a quality tradeoff for fast quantifiable prototyping and 2D design.

Pipeline: As in Figure 1, our pipeline starts with a scanned hand-drawn sketch, which is converted to a set of (almost cubic) Bézier curves (using this method [FLB16]). These curves are then post-processed (an engineering task that can be easily automated) to create a set of connected tangent continuous curve segments. These segments might contain multiple and open curves, T-junctions, and intersections, which our sampling algorithm can handle gracefully by limiting the minimum distance between samples (see paragraph Robustness below). The curves are then fed to our ε-curve sampler, and the edges between resulting adjacent samples can then be used as constraining edges for any meshing algorithm (we used CGAL mesher [RY07] based on Shewchuk's meshing algorithm [She00]). Please note that though the recent TriWild meshing [HSG*19] can also take Bézier curves as input, output non-linear triangles, and offers good performance in triangle quality, it requires a fixed target edge length, resulting in a dense uniform mesh while considering small features. Moreover, we focus on interactive approximation meshing, and hence we restrict our comparison to the CGAL mesher which can also output nonuniform meshes.

Finally, our pipeline is completed by running a common finite element simulation on this mesh that aims to interpolate data given at the input curves as smoothly as possible. In this simulation, we discretize the Poisson problem using the well-known 'cotan' Laplacian over the triangulation. We apply Dirichlet conditions of equal

value on the vertices corresponding to the sampled input curve as well as on the rectangular boundary. This simulation illustrates the trade-off between the number of triangles, precision, and runtime, and can be applied to common Poisson problems in physics, including fluid pressure, static electrical fields, soap films (and other minimal surfaces), heat conduction, etc.

Robustness: One of the challenges in curve reconstruction and sampling is handling sharp features along which the local feature size vanishes. We tackle this by specifying a minimum distance d_{min} between samples that captures the size of the smallest desired feature. During the sampling procedure, we simply need to check whether the distance to the previous sample is at least d_{min}; otherwise, we step along the foot points on the curve until d_{min} (in our experiments, 0.01 of bounding box diagonal) is reached.

Required ε: An $\varepsilon = 0.66$ parameter for sampling the curves not only guarantees the reconstruction of the original curve by CompatibleCrust but also a conforming Delaunay triangulation when meshing, and subsequently a positive Jacobian for triangles with non-linear boundary edges. However, since our simulation is approximative, we can use an even more relaxed $\varepsilon = 1$ that generates fewer samples and, subsequently, many fewer triangles in the triangle mesh. Figure 12 shows three of the five sketches used in the evaluation (except Lion and Croco, see Figures 1, 9), together with Bézier curves, sampled polylines, and triangle count meshes for a minimum sampling density of 0.1 of bounding box diagonal.

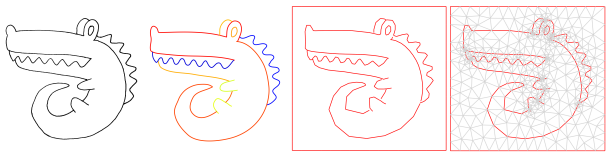

Figure 9: *From left to right: Scanned user sketch, after fitting Bézier curves, sampled with* $\varepsilon = 1$ *constrained to [0.01, 0.1] distance in terms of the bounding box diagonal, and the result of feature-aware meshing with max edge length = 0.1.*

Tradeoff evaluation: In order to show the speed/quality tradeoff for simulations on the meshed triangulations, we evaluate the mesh on five sketches that we sample with six varying sampling densities (0.1, 0.05, 0.03, 0.02, 0.015, and 0.01), each using the densest (the last one) as the baseline. Figure 13 shows the six polylines for varying sampling densities, the resulting meshes, and their respective simulation results for the Lion sketch. While generating samples, we limit the sampling density with a parameter d_{max} that we enforce similarly to d_{min}, but this time by backtracking the foot points on the curve. In the appendix, Table 4 lists triangle count, simulation runtime, and RMSE with respect to the baseline averaged over the five sketches (Figures 12, 13) for the six sampling densities, these results are visualized in Figure 10 for uniform and non-uniform meshing respectively. For uniform meshing, runtime goes up almost quadratically with the number of triangles (and therefore fourth-order with sampling density), while reducing the simulation time by two orders of magnitude only doubles the error. For non-uniform meshing, runtime is much more linear but exhibits a higher error, due to many fewer triangles in the mesh. The total runtime for our sampling (with $\varepsilon = 1$), (with reduced N=100), meshing (sampling density 0.03), and simulation steps for these sketches is less than a second, which offers an interesting tradeoff between speed and quality so that changes to the input can be roughly simulated in interactive time, a promising feature for sketch-based design.

6. Discussions and Future Work

We proposed an algorithm that ε-sample curves (based on a parameter ε) in a quantified and feature-aware manner.The proposed computational framework is based on an efficient estimate of the local feature size of the points of the curve, which only requires a set of one-sided polylines, approximating branches of the medial axis (seen from two sides of the curve). The precision of this approximation can be controlled arbitrarily close. We, however, assume a prior lower bound for the reach of the curve, which is a natural requirement for any discrete representation of smooth curves. One may see that as the main limitation of our proposed sampling algorithm. However, since the exact lfs computation in a discrete setting is an ill-posed problem by definition, we do not see any alternatives to our lfs approximation strategy.

We also evaluated the corresponding approximation error as a tradeoff with the acceptable computation budget for sampling. To highlight the utility of such an efficient while quantifiable sampler, we implemented the theoretical state-of-the-art reconstruction method CompatibleCrust and compared its reconstruction performance based on a curve reconstruction benchmark. Remarkably, we managed to reduce the theoretical upper bounds for ε in the context of ε-sampling, to ensure reconstruction guarantees in practice for a number of algorithms. We achieved this by brute-force searching many point configurations in order to find counter-examples for specific ε values. This study opens new lines of research in the context of curve reconstruction by evaluating the quality of the known theoretical bounds on ε, and comparing them with the practical performance of existing methods for the first time under quantifiable sampling settings.

Our proposed sampling framework can be extended in various ways, e.g., be adapted to ρ-sampling [OMW16], or combine feature size-based distances with absolute distance limits between the original smooth curve and its sampled approximation. Other types of curves can be integrated into our framework easily, such as lower order quadratic Bézier curves, circular arcs, lines, or even NURBS. Finally, the extension of our framework to an ε-sampler for surfaces in 3D seems to be a promising and important future work avenue, having applications, e.g., in determining upper bounds for ε-sampling guarantees which are very weak in 3D, mesh simplification, and meshing for finite element simulations.

7. Acknowledgements

We thank David Hahn for providing Laplacian matlab simulation code. This work has been partially funded by the Austrian Science Fund (FWF) project no. P32418-N31 and by the Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF) project ICT19-009.

References

[AAA*09] AICHHOLZER O., AIGNER W., AURENHAMMER F., HACKL T., JÜTTLER B., RABL M.: Medial axis computation for planar free–form shapes. *Computer-Aided Design 41*, 5 (2009), 339–349. 2

[ABE98] AMENTA N., BERN M., EPPSTEIN D.: The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing 60*, 2 (1998), 125–135. 1, 3

[ABE09] ATTALI D., BOISSONNAT J.-D., EDELSBRUNNER H.: Stability and computation of medial axes-a state-of-the-art report. *Mathematical foundations of scientific visualization, computer graphics, and massive data exploration* (2009), 109–125. 2

[BB22] BAKKE BJERKEVIK H.: Tighter bounds for reconstruction from ε-samples. In *38th International Symposium on Computational Geometry (SoCG 2022)* (2022). 2, 3, 5

[Blu67] BLUM H.: A Transformation for Extracting New Descriptors of Shape. In *Models for the Perception of Speech and Visual Form*, Wathen-Dunn W., (Ed.). MIT Press, Cam., 1967, pp. 362–380. 2

[DK99] DEY T. K., KUMAR P.: A simple provable algorithm for curve reconstruction. In *SODA* (1999), vol. 99, pp. 893–894. 3

[DMR00] DEY T. K., MEHLHORN K., RAMOS E. A.: Curve reconstruction: Connecting dots with good reason. *Computational Geometry 15*, 4 (2000), 229 – 244. 3

[DW01] DEY T. K., WENGER R.: Reconstructing curves with sharp corners. *Computational Geometry 19*, 2 (2001), 89 – 99. Combinatorial Curves and Surfaces. 3

[DW02] DEY T. K., WENGER R.: Fast reconstruction of curves with sharp corners. *International Journal of Computational Geometry & Applications 12*, 05 (2002), 353–400. 3

[DYH*20] DAS A., YANG Y., HOSPEDALES T., XIANG T., SONG Y.-Z.: Bézier sketch: A generative model for scalable vector sketches. In *Computer Vision–ECCV 2020* (2020), Springer, pp. 632–647. 2

[eig] https://eigen.tuxfamily.org/. [Online; accessed 05-June-2023]. 5

[Fed59] FEDERER H.: Curvature measures. *Transactions of the American Mathematical Society 93*, 3 (1959), 418–491. 3

[FLB16] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–10. 1, 6, 9

[HSG*19] HU Y., SCHNEIDER T., GAO X., ZHOU Q., JACOBSON A., ZORIN D., PANOZZO D.: TriWild: robust triangulation with curve constraints. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–15. 6

[Len06] LENZ T.: How to sample and reconstruct curves with unusual features. In *EWCG: Proc. of the 22nd European Workshop on Computational Geometry* (2006), pp. 29–32. 3

[MOW22] MARIN D., OHRHALLINGER S., WIMMER M.: SIGDT: 2D curve reconstruction. *Computer Graphics Forum 41*, 7 (2022), 12. 3

[nan] https://github.com/memononen/nanosvg. [Online; accessed 05-June-2023]. 9

[OM13] OHRHALLINGER S., MUDUR S.: An efficient algorithm for determining an aesthetic shape connecting unorganized 2D points. In *Computer Graphics Forum* (2013), vol. 32, pp. 72–88. 3

[OMW16] OHRHALLINGER S., MITCHELL S. A., WIMMER M.: Curve reconstruction with many fewer samples. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 167–176. 2, 3, 4, 5, 7

[OPP*21] OHRHALLINGER S., PEETHAMBARAN J., PARAKKAT A. D., DEY T. K., MUTHUGANAPATHY R.: 2D points curve reconstruction survey and benchmark. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 611–632. 3, 4, 5, 6

[PM16] PARAKKAT A. D., MUTHUGANAPATHY R.: Crawl through neighbors: A simple curve reconstruction algorithm. In *Computer graphics forum* (2016), vol. 35, Wiley Online Library, pp. 177–186. 3

[PMC22] PARAKKAT A. D., MEMARI P., CANI M.-P.: Delaunay painting: Perceptual image colouring from raster contours with gaps. In *Computer Graphics Forum* (2022), vol. 41, pp. 166–181. 1, 6, 11

[PMM18] PARAKKAT A. D., METHIRUMANGALATH S., MUTHUGANAPATHY R.: Peeling the longest: A simple generalized curve reconstruction algorithm. *Computers & Graphics 74* (2018), 191 – 201. 3

[PS18] PAGANI L., SCOTT P. J.: Curvature based sampling of curves and surfaces. *Computer Aided Geometric Design 59* (2018), 32–48. 2

[qui] https://github.com/ZhepeiWang/Root-Finder. [Online; accessed 05-June-2023]. 9

[RG03] RAMANATHAN M., GURUMOORTHY B.: Constructing medial axis transform of planar domains with curved boundaries. *Computer-Aided Design 35*, 7 (2003), 619–632. 2

[Rup93] RUPPERT J.: A new and simple algorithm for quality 2-dimensional mesh generation. In *Proc. 4th ann. ACM-SIAM SODA* (Philadelphia, PA, USA, 1993), SODA '93, SIAM, pp. 83–92. 2

[RY07] RINEAU L., YVINEC M.: A generic software design for Delaunay refinement meshing. *Computational Geometry 38* (2007), 100–110. 6

[Sch07] SCHUMAKER L.: *Spline functions: basic theory*. Cambridge university press, 2007. 2

[SDC09] SỲKORA D., DINGLIANA J., COLLINS S.: Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 599–608. 6

[She00] SHEWCHUK J. R.: Mesh generation for domains with small angles. In *Proceedings of the sixteenth annual Symposium on Computational Geometry* (2000), pp. 1–10. 6

[WC20] WOŹNY P., CHUDY F.: Linear-time geometric algorithm for evaluating bézier curves. *Computer-Aided Design 118* (2020), 102760. 2

[YBM04] YANG Y., BROCK O., MOLL R. N.: Efficient and robust computation of an approximated medial axis. In *Symposium on Solid Modeling and Applications* (2004), pp. 15–24. 2

Supplementary Material

7.1. Implementation Details for Cubic Bézier Curves

Our sampler is implemented for piece-wise cubic Bézier curves as it provides a flexible and computationally efficient representation. In our experiments, along with manually created curves using Inkscape and parsed using open source nanosvg [nan], we also used scanned hand-drawn sketches that are vectorized using [FLB16] since curve identification is not part of our contribution. Our sampling does not require any tangent continuity between the piece-wise cubic Bézier curve segments. However, using C^1 continuity will avoid the vanishing of the local feature size at joints, leading to better results, though, for our applications, we show that we can handle these artifacts as well.

The sampling process on Bézier curve segments is performed by inserting foot points as follows. Let $p \in C$ correspond to a point evaluated as $B_k^3(t)$ on curve segment k at parameter value t; for simplicity, we will denote this using the notation $B(t)$. Each foot point $f_i \in C$ can be evaluated as $f_i = B(t)$. The normal $n(t)$ to C at a point $p = B(t)$ is orthogonal to the tangent $B'(t)$ at p, thus $n(t) = [B'_y(t), -B'_x(t)]^T$. To compute the medial point m_i for a foot point $f_i \in C$, we determine m_i as the center of a disk D_i with radius r_i. This disk is tangent to C at f_i and constrained by passing through the subsequent foot point f_{i+1}, thus fixing r_i and, later, m_i. As m_i may be located on either side of the curve, we first initialize our radius estimate r_i with a circle of radius equal to the bounding box diagonal of C. For both sides, we then compute the closest point q of the curve (for all segments) to the current medial point estimate, and if it is closer, replace it with the center of the circle tangent to C at f_i and passing through q. We iterate until a desired precision threshold (10^{-9}) is reached and select the point m_i from the side that is closer to f_i. Note that this threshold always converged in our experiments but could be further adjusted proportionally to curve sampling distances.

The distance of a point m to a cubic Bézier curve B can be expressed as $|B(t) - m|$. We translate the coordinates of B such that m is at the origin. Then, in order to find the point on B closest to m, $B(t)$, we minimize the squared term $|B(t)|^2$, thus eliminating the square root, by setting its first derivative $(|B(t)|^2)' = 0$. This results in a quintic polynomial that we solve using a specialized quintic root finder [qui] that employs the real roots isolation method using both Cauchy's bound as well as Kojima's bound, as it is several times faster than the general Eigen library method (see Sec. 3.4 for the analysis). Since computing the closest point to a cubic Bézier curve is still expensive, we first test its bounds using its convex hull property. Testing whether any point of a convex hull edge is closer than the current minimum distance, using the same edge projection procedure as in Sec. 3.2, accelerates it further, resulting in a global runtime reduced by an order of magnitude, as shown in Table 2.

Uniform meshing						
Density	0.1	0.05	0.03	0.02	0.015	0.01
Triangles	1087	1654	3373	6702	11333	24536
Runtime	0.011	0.021	0.069	0.273	0.835	6.784
RMSE	3.526	3.106	2.700	1.933	1.703	-
Uniform meshing						
Triangles	249	252	267	301	350	477
Runtime	0.013	0.012	0.014	0.013	0.016	0.074
RMSE	7.631	3.481	2.039	2.354	1.268	-

Table 4: *Sampling densities evaluated for the simulation w.r.t. 0.01: Density is maximum sampling distance/edge length in bounding box diagonal, runtime in seconds, and RMSE in terms of the uniform Dirichlet boundary conditions, for the five sketches' average.*

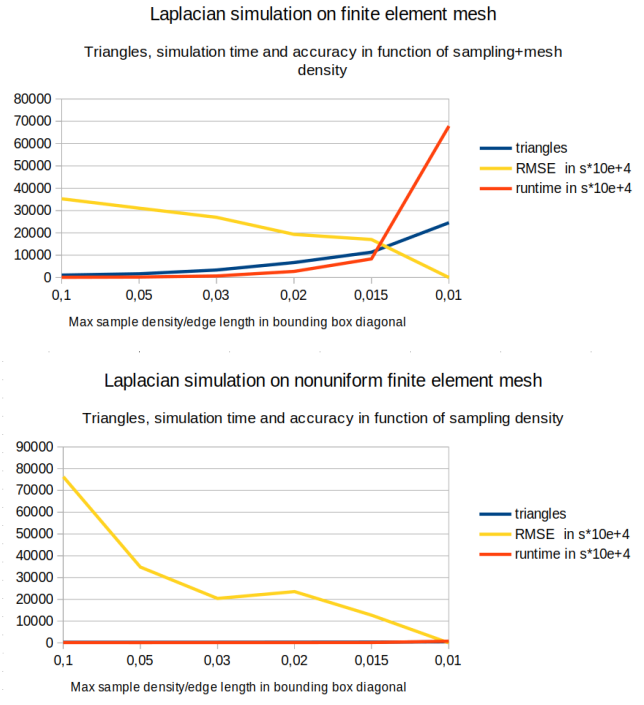

Figure 10: *For varying sampling densities (=maximum edge length of the meshed triangulation), triangle count, runtime and RMSE (compared to 0.01 density) of the Laplacian simulation are shown.*

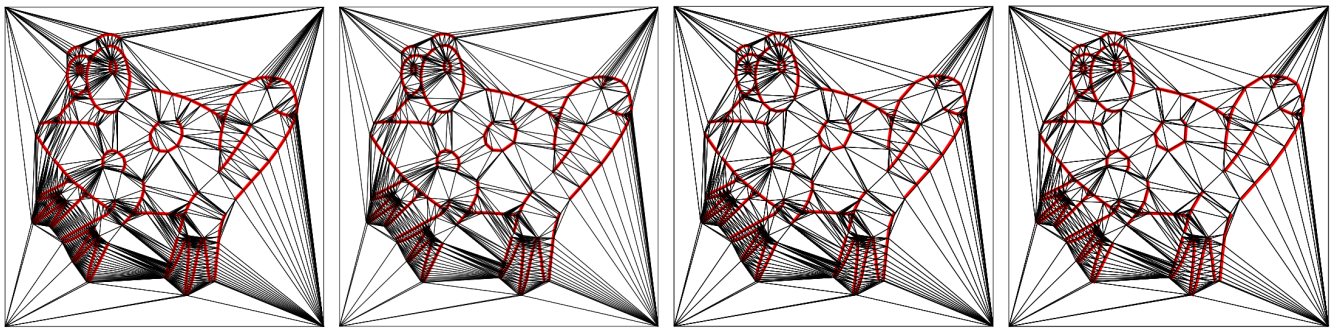

Figure 11: *Delaunay conforming property with different* ε *values. Left to right:* $\varepsilon = 0.4, 0.5, 0.6, 0.7$

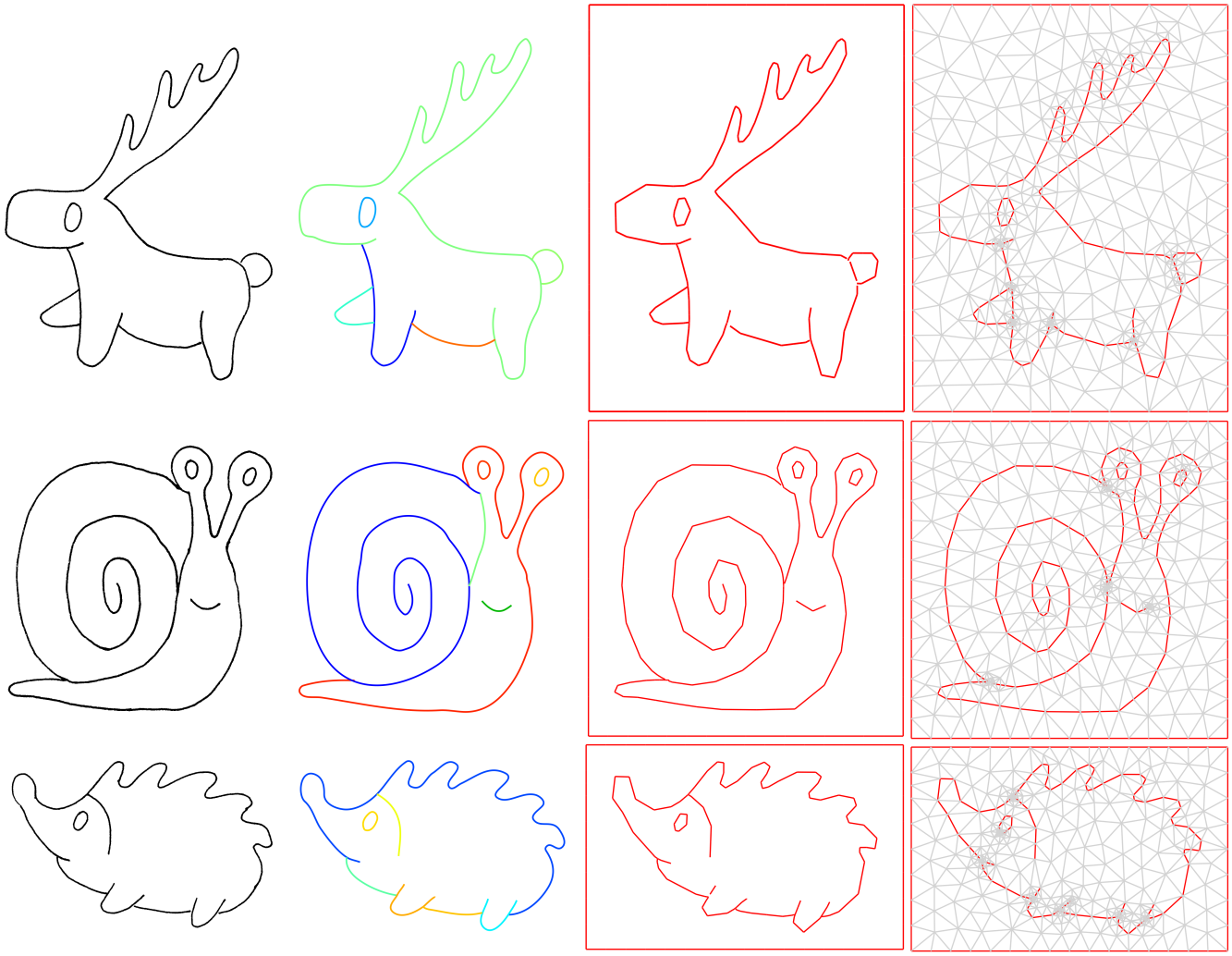

Figure 12: *From left to right: Scanned user sketch, after fitting Bézier curves, sampled with* $\varepsilon = 1$ *constrained to [0.01, 0.1] distance in terms of the bounding box diagonal, and the result of feature-aware meshing with max edge length = 0.1.*

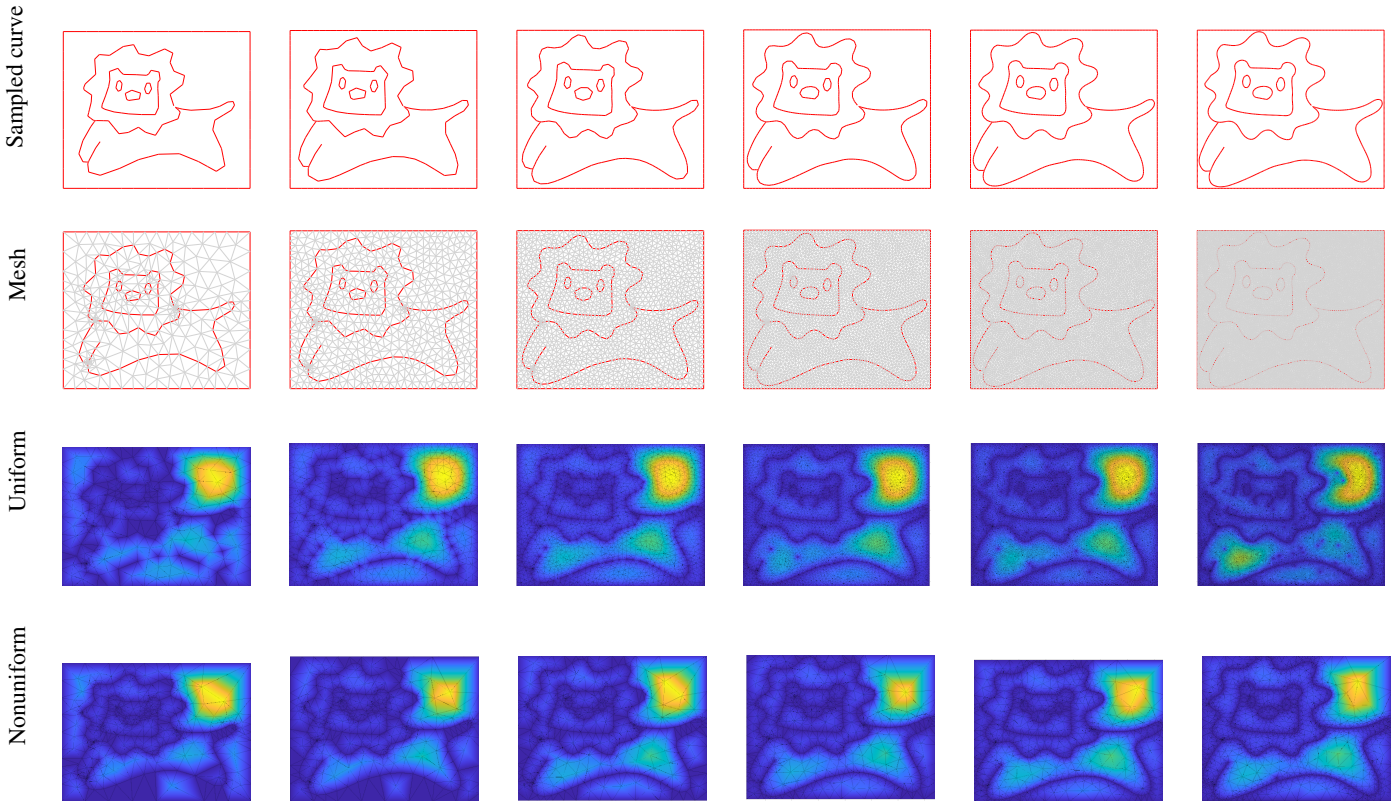

Figure 13: *Left to right: Varying sampling densities of 0.1, 0.05, 0.03, 0.02, 0.015, and 0.01 of bounding box diagonal for the* LION. *Top to bottom: Sampled curve connected by polylines, Meshed triangulation, and Visualization of the Laplacian simulation results with the Dirichlet conditions as dark blue dots from the vertices of the above polylines as well as the rectangular boundary, above with uniform and then below with non-uniform meshing.*

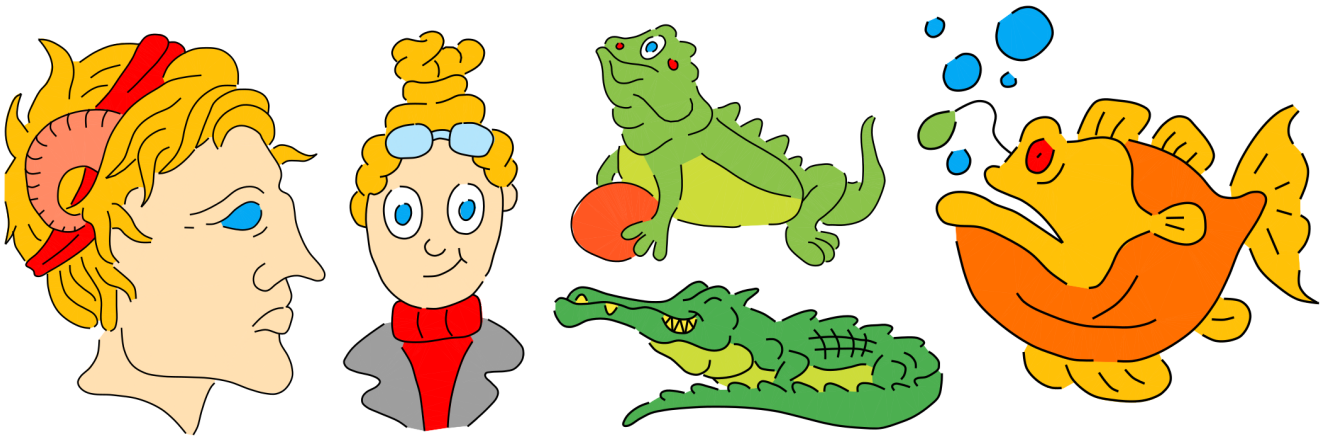

Figure 14: *Vector sketches colored using our improved Delaunay Painting [PMC22] - Images taken and vectorized from [PMC22]*