EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Adapting Feature Curve Networks to a Prescribed Scale

Anne Gehre Isaak Lim Leif Kobbelt

Visual Computing Institute, RWTH Aachen University

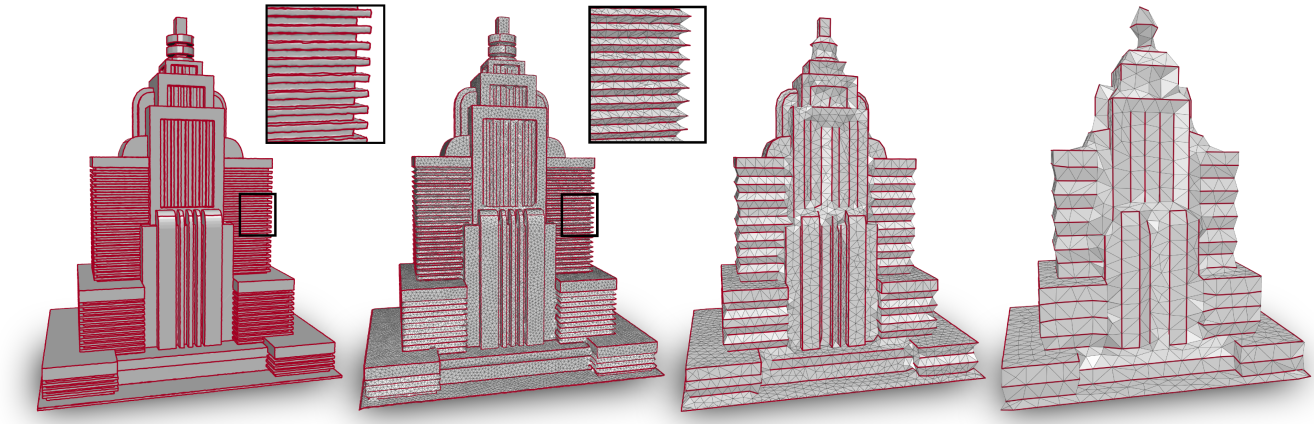


Figure 1: *Feature curve networks (red edges) generated with our approach are adapted to different scales (from fine to coarse). Our goal is to preserve as much feature information as possible while avoiding features that are too close to each other for a given target edge length. This requires global optimization and cannot be solved by greedy approaches or combination of filtering and thresholding. Here, the feature curves are used as boundary constraints for isotropic remeshing at different target edge lengths. The input-mesh with the initial feature curves is shown on the left.*

Abstract

Feature curves on surface meshes are usually defined solely based on local shape properties such as dihedral angles and principal curvatures. From the application perspective, however, the meaningfulness of a network of feature curves also depends on a global scale parameter that takes the distance between feature curves into account, i.e., on a coarse scale, nearby feature curves should be merged or suppressed if the surface region between them is not representable at the given scale/resolution.
In this paper, we propose a computational approach to the intuitive notion of scale conforming feature curve networks where the density of feature curves on the surface adapts to a global scale parameter. We present a constrained global optimization algorithm that computes scale conforming feature curve networks by eliminating curve segments that represent surface features, which are not compatible to the prescribed scale. To demonstrate the usefulness of our approach we apply isotropic and anisotropic remeshing schemes that take our feature curve networks as input. For a number of example meshes, we thus generate high quality shape approximations at various levels of detail.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Computational Geometry, Digital Geometry Processing, and Level Of Detail Algorithms—Feature Curve Networks

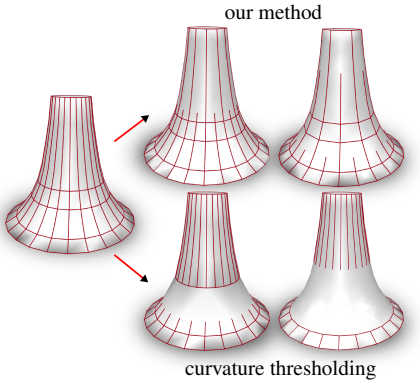
1. Introduction

Acquisition techniques for 3D geometry have improved vastly and the use of 3D digital models has found its way into many application areas e.g., simulation, medical applications, or visualization. Usually, the acquired raw data requires further geometric processing to tackle problems such as bad element quality, noise, or large storage requirements. Various processing techniques exist that establish good element quality (e.g. [ECBK14,CJL11,BZK09,BK04,

ADVDI03]). The surface approximation is considered high quality if elements have a high regularity and align to surface features.

These features or guiding constraints need to be computed in advance and are given as input to the respective methods. Current feature curve extraction approaches (e.g. [NSP10, WG09, YBS05, OBS04]) take local shape properties into account, where the number of selected feature curves is controlled by filtering or by setting thresholds on curvature values.

Typically, remeshing algorithms are provided with a user-specified target edge length or complexity. Here, applications for enforcing complexity constraints can range from noise suppression, to removing small scale details (e.g. to reduce computation times in simulations), or to depicting up to very coarse resolutions of the mesh (e.g. when the image resolution or available storage is low).



However, none of the previously mentioned feature extraction algorithms take the spatial feature density into account. Suppose there are two feature curves nearby, one with a relatively high "sharpness" and the other one with lower. Then a filter and threshold approach would be able to suppress the less sharp feature. At the same time, however, it would also unnecessarily suppress other feature curves with the same (lower) sharpness even if they are not close to another feature (see image above). This example clearly shows that feature selection is a global optimization problem and the decision to suppress a feature should be dependent on the existence of other features in the vicinity. Hence, using the aforementioned mechanisms we cannot preserve the maximal set of feature curves representable at a given resolution.

Therefore, we propose a feature curve adaption mechanism as a novel fully automatic alternative to this problem. Our approach adapts feature curve networks (FCNs) to a prescribed scale by looking at the network as a whole and not treating each feature curve separately, i.e., we preserve all features which can be represented at the given target resolution irrespectively of their sharpness.

Contribution In this paper we propose an algorithm that adapts feature curve networks to a given scale. Our main contributions can be summarized as follows:

- We present a computational approach to the intuitive notion of a scale conforming feature curve network in Section 4.
- We propose an efficient method for the computation of scale conforming feature curve networks, which preserve the maximum set of prominent feature curves within a prescribed scale from an initial set of feature lines. Details are given in Section 5.
- As proof of concept we demonstrate the usefulness of scale conforming feature curve networks by generating meshes at several resolutions that preserve the detected features at different scales. Results are discussed in Section 6.

2. Related Work

Feature Line Detection A large body of work on detection of feature lines in a 3D model exists. It is out of scope of this paper to present an entire survey on feature extraction (see [LZH*07, Dem09] for more details), hence we will only describe a few representatives. Several different approaches based on the detection of ridges and ravines as curvature extrema of a surface exist. While [BPK98, OBS04] extract these from an implicit surface representation, [WB01] find curvature extrema by detecting triangle degeneracies on focal surfaces. [YBS05] trace crest lines by employing polynomial fitting in order to estimate curvatures. Weinkauf and Günther [WG09] detect feature lines from the minimal- and maximal curvature fields of a mesh surface by connecting maxima and saddle points to form a feature skeleton. By iteratively removing saddle points, they can coarsen this skeleton. Nieser et al. [NSP10] grow patches around seeds where the maximal curvature of a set of neighboring faces lies below a predefined threshold. [CYW14] also takes the curve length (but not the feature density) into account by filtering a dense set of features for salient and long curves.

A further possibility for the generation of feature curves is to extract patch boundaries from segmentation algorithms based on normal clustering. Here, the amount of curves is related to the number of clusters. Variational Shape Approximation (VSA) [CSAD04] partitions faces based on Lloyd's clustering. Lai et al. [LZHM06] present a segmentation based on mean-shift clustering. Zhuang et al. provide an interactive partitioning, where a user sets seed points for "live-wires", which align along curvature directions [ZZCJ14].

Existing approaches perform feature suppression by either smoothing the object surface in advance, to suppress small scale details, or by thresholding curvature. Both mechanisms are based on local surface properties. None of these approaches take the distance between feature curves into account. In contrast, we propose a global optimization procedure that identifies a maximal set of feature edges that are representable at a given target resolution.

Shape Abstraction and Approximation Via Curve Networks A set of abstraction mechanisms exist which exploit feature curve networks for shape approximations. Sala et al. [SD08] developed a 2D abstraction of contour edges by generating multi-scale images, which are based on a user specified part-vocabulary. Mi et al. [MDS09] abstract 2D shapes by decomposing them into perceptual parts. For abstraction they remove parts based on user-specified thresholds and reconstruct the shape from the remaining ones.

Mehra et al. [MZL*09] generate hierarchical abstraction curve networks for man-made objects. The object surface is approximated by fitting the outer hull of a voxel grid to it. Curve networks are then extracted with VSA and simplified by thresholding the reconstruction error, based on the curve smoothness and the normal deviation of the resulting mesh approximation. De Goes et al. [DGGDV11] define the concept of an Exoskeleton where they generate a segmentation of the surface, revealing meaningful perceptual parts, which are refined by employing VSA. A user can further influence the amount of generated curves by adding seeds for VSA.

In contrast to existing approaches our goal is to generate approximations of various kinds of shapes (not only man-made), where the level of detail is clearly defined by a prescribed scale parameter.

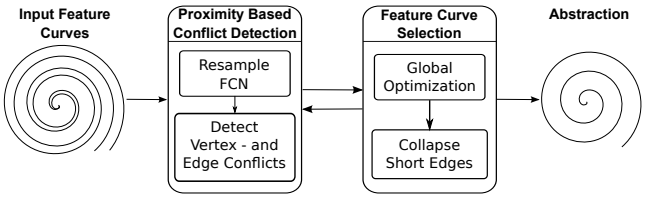


Figure 2: *Overview over our feature curve network adaptation scheme: As input we receive an initial feature network. In the* ***proximity based conflict detection*** *we check whether the network conforms to the prescribed scale. For this we have to find a discrete representation via resampling. We can then describe scale-conformity as vertex and edge conflicts. In the* ***feature curve selection*** *the detected conflicts are resolved. Since edges are weighted according to their relevance, we resolve conflicts between edges by suppressing the less significant edges in a global optimization step. Remaining edges that are shorter than the minimal scale are collapsed. This can change the network topology, which requires further conflict detection. We iterate this process until no conflicts exist and the transformed feature network conforms to the prescribed scale.*

Triangulations with Feature Curves Ruppert et al. present an approach to generate 2D triangulations around planar shape line graphs which are guaranteed to have a prescribed minimum inner angle [Rup95]. By only constraining angles we cannot make any assumptions on the curve density in our formulation. Hence, we present a similar idea in the formalization of a scale conforming FCN, only that we constrain the minimum edge length, to restrict the curve density. Details are given in Section 4.

3. Problem Statement

Conceptually, our scale adaptation approach is quite different from previous ones. Existing approaches that are based a (pre-)filter operation eliminate features by locally blurring the sharpness until it falls below a given threshold. Hence, the decision whether a feature curve is eliminated is made just locally without considering other features in the vicinity. This leads to situations where more features than necessary are removed since even a less pronounced feature (which is smoothed out by the filter) could be valuable on a given scale if neighboring features are sufficiently far away. In contrast to this local filter approach, we are following a globally optimal sub-sampling approach where feature curves are eliminated only if there is another (more relevant) conflicting feature curve nearby.

Given an initial set of feature curves on a surface mesh $\mathscr{M}$ we generate abstractions in form of FCNs, which are guaranteed to comply with a prescribed global proximity constraint. This constraint is given in form of a scale parameter $r_{\min}$, which limits the target resolution (e.g. the minimum edge-lengths of the respective approximation). To extract the final FCN we follow the workflow depicted in Figure 2, which consists of two main building blocks.

The first module involves a *proximity based conflict detection* procedure, which identifies curve segments that cannot coexist at the given scale. In this continuous setting finding appropriate segments is intractable (because there exist infinitely many possible segments). Thus, we discretize this problem by introducing the definition of a *scale conforming* FCN in Section 4, which resamples feature arcs into scale conforming segments. This discretization allows for the identification of conflicting curve segments (which we denote as edges of the FCN in the following) and the formulation of an objective function measuring the quality of the edges.

The identified set of conflicting features serves as input to the second component: the *feature curve selection*. Feature selection is performed with the goal to best preserve the structure of the initial network. In particular we resample features such that the geometric error is minimal and remove a minimal set of edges so that the remaining (non-conflicting) edges preserve as much as possible of the original feature network.

Due to the discrete representation of the feature curves we are able to evaluate the relevance of each edge. Since the requirements imposed upon a feature curve network are highly application dependent, we allow the user to combine different quality criteria. The first set of criteria evaluates the quality of a *single* edge based on its local properties: across edge *sharpness*, *curve length*, *symmetry*, and whether the feature edge belongs to a closed *loop*. Secondly, we also regard *combined* quality criteria, which rate pairs of edges according to their *smoothness*, *orthogonality*, and *parallelism*. Exact details will be described in Section 5.3.1. Depending on the application any other combination of criteria can also be integrated into our approach (e.g. different saliency based edge weights).

For *feature curve selection*, the described quality criteria are incorporated in a functional. In a global optimization scheme, we select those curve segments of the FCN which maximize this function, while abiding proximity constraints.

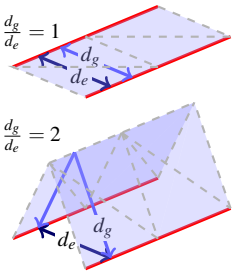
4. Scale Conforming Feature Curve Networks

State of the art remeshing algorithms produce high quality surface approximations based on a set of features or guiding constraints which is either precomputed or provided as input. Moreover, a target scale can be prescribed to control the mesh complexity. In order to abide to such a density constraint, mesh elements can only align along a subset of surface features which are not contradictory to the prescribed target edge length. We denote such a subset as a *scale conforming FCN*. In the following we will define FCNs and describe conditions that they must fulfill to achieve scale conformity.

Definition 1 (FCN) For a given mesh $\mathcal{M}$ we define a *feature curve network* as a 3-tuple $N = (V, E, A)$ where the vertices $v \in V$ lie on the continuous surface defined by $\mathcal{M}$ but not necessarily at a vertex of $\mathcal{M}$. According to their valence in N, we split the set of vertices V into two disjoint subsets $\bar{V}$ and V^* that contain the regular vertices (valence $= 2$) and the extraordinary vertices (valence $\neq 2$) respectively. E is the set of edges which connect two vertices in the feature network. The set A consists of (mutually disjoint) arcs. An arc a is a sequence of feature edges that either connects two extraordinary vertices or forms a closed loop.

The input scale parameter $r_{\min}$ provides a lower bound for the sampling rate across the continuous surface defined by $\mathcal{M}$. Hence, a remeshing procedure should adapt the vertex positions to this

lower bound such that all edges are longer than $r_{\min}$. Accordingly, we denote an FCN as *scale conforming* to a given scale $r_{\min}$ if it can be extended to a manifold triangle mesh $\mathcal{N}$ (by adding vertices and edges) that fulfills the following conditions:



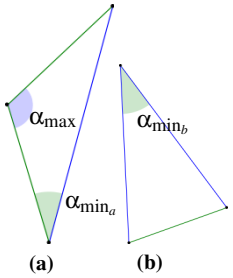
1. Each edge e of $\mathcal{N}$ has its length $||e|| \geq r_{\min}$.
2. The ratio of the geodesic distance d_g in $\mathcal{N}$ to the Euclidean distance d_e between nearby feature curves is bounded by some $q < 2$. This condition ensures that the surface $\mathcal{N}$ approximates $\mathcal{M}$ well, avoiding high amplitude surface oscillations between feature curves. E.g. the image above shows two triangulations between the (red) feature curves. In contrast to the upper triangulation the lower one introduces implausible detail.

This definition of a scale conforming FCN serves as a conceptual goal that the network should achieve, according to which we derive conflict detection and suppression schemes for feature edges in the following. These are required to find a maximal subset of feature curves, which are scale conforming.

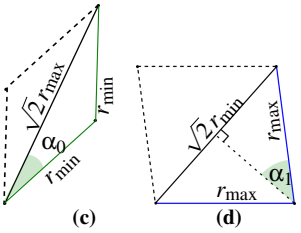
4.1. Quality Criteria

Mesh Quality The triangles that are adjacent to the scale conforming FCN should have sufficient quality. If we allow arbitrarily long edges, degenerate triangle configurations can occur. We can limit their aspect ratio by providing an upper bound $r_{\max}$ on the edges of the scale conforming FCN, bounding aspect ratios to $\left[1, \frac{r_{\max}}{r_{\min}}\right]$. Thus, to ensure triangle quality we define a scale interval $S = [r_{\min}, r_{\max}]$ so that all edges of the FCN have their length in S.

Feature Edge Angles The upper and lower bounds on the edge lengths also impose upper and lower bounds $[\alpha_{\min}, \alpha_{\max}]$ on the inner angles of each triangle (see image below). Consequently, each pair of adjacent edges in E has to enclose an angle from $[\alpha_{\min}, \alpha_{\max}] \cup [2\alpha_{\min}, 2\alpha_{\max}] \cup [3\alpha_{\min}, 3\alpha_{\max}]$. This set of feasible angles may consist of one, two, or three intervals, depending on whether $\alpha_{\max} < 2\alpha_{\min}$ or even $2\alpha_{\max} < 3\alpha_{\min}$. The smallest maximal angle in a triangle is $\frac{\pi}{3}$. Hence, $3 \cdot \alpha_{\max} \geq \pi$ is the maximal possible angle between two edges, so we do not need to consider any further intervals. Depending on the scale interval S the smallest representable angle $\alpha_{\min}$ is either found in triangle type (a) with $\alpha_{\min_a} = \arccos(r_{\max}/2r_{\min})$ or in triangle type (b) where $\alpha_{\min_b} = 2 \cdot \arcsin(r_{\min}/2r_{\max})$. If $r_{\max}/r_{\min} < 1.618$ the angle in triangle (b) is smaller.



Feature Edge Angles for Quad Meshes We can further extend this definition to apply to quad meshes. In this case the diagonal should not be represented in the FCN. Therefore, the minimal and maximal angles have to be adapted in the following way: Since we want to constrain the diagonal to have a length in the interval $[\sqrt{2}r_{\min}, \sqrt{2}r_{\max}]$, we can derive the minimal and maximal angles from the two configurations shown to the right. Thus, the minimal angle is $\alpha_{\min} = 2\min(\alpha_0, \alpha_1)$. The maximal angle can be derived analogously by switching $r_{\min}$ and $r_{\max}$ in configurations **(c)** and **(d)**.



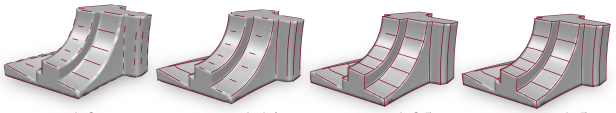


Figure 3: *Effect of constraining FCNs to different aspect ratios for the fandisc at the same scale. The selected aspect ratios $\frac{r_{\max}}{r_{\min}}$ are given below. Aspect ratios close to one constrain angles of adjacent features to be close to multiples of 60 degrees. We found aspect ratios of 1.5 to give a good tradeoff between triangle shape and representable angles. For values greater than 1.5 no further changes would be visible in this example.*

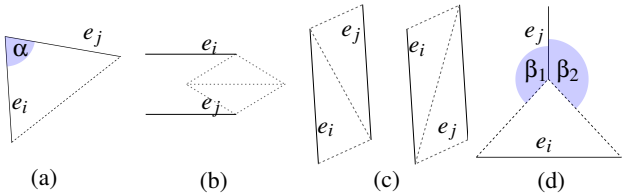


Figure 4: *Conflict detection: (a) shows the case where two edges e_i and e_j share a common node. By testing the angle α between them it is possible to determine whether they are in conflict. (b), (c), and (d) show possible triangulations of non-adjacent edges e_i and e_j.*

Aspect Ratio The range S should be chosen such that a sufficient quality of the output mesh $\mathcal{N}$ is guaranteed while not being to rigidly confined to (close to) equilateral triangles. Figure 3 demonstrates the effect of constraining the upper bound of edge-lengths in abstractions of the fan-disc model. The first two FCNs with aspect ratios of $\frac{r_{\max}}{r_{\min}}$ close to one lead to gaps in the feature lines because the angles between the line-segments are not representable in the angle intervals. For ratios $\frac{r_{\max}}{r_{\min}} > 1.365$ the representable angle intervals merge (ie. $\alpha_{\max} < 2\alpha_{\min}$), which allows to represent a broad range of angles. In practice we found that $\frac{r_{\max}}{r_{\min}} = 1.5$ is a good choice since it includes 'triangulated quad meshes' in the set of acceptable output meshes (because the lengths of the diagonals in a triangulated quad mesh are about $\sqrt{2}$ times the lengths of the quad edges).

4.2. Achieving Scale Conformity

The definition of a FCN conforming to a given scale gives rise to a number of conditions that N has to satisfy, especially when fulfilling the quality criteria discussed above. First of all, N conforming to $r_{\min}$ trivially implies that all edges in E have to be longer than $r_{\min}$. Furthermore, constraining the aspect ratio of the triangles that are adjacent to E implies that all edges have their length in S.

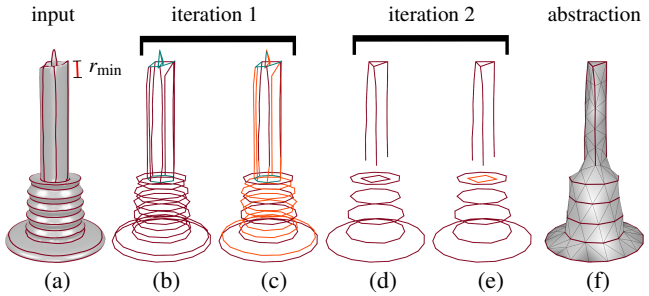


Figure 5: *Intermediate steps of the FCN abstraction.* ***(a)*** *Input.****(b)*** *First resampling with red curves in the prescribed lengths, cyan curves are short curves, which will induce vertex conflicts.* ***(c)*** *After proximity based conflict detection and optimization. Orange curve segments are marked as 'delete' by the optimization due to conflicts with other curves.* ***(d)*** *Conflicts are removed and the arcs are resampled.* ***(e)*** *Resampled arc set that was formerly composed of short arcs introduces new conflicts, which are removed (orange).* ***(f)*** *Final abstraction.*

Different conditions need to apply for *adjacent* and *non-adjacent* edge pairs:

In order to adjust to the prescribed quality criteria, angles between *adjacent* edge-pairs need to be in the described bounds (cf. Figure 4a). Also, edge-pairs that are connected by a short edge e_s ($||e_s|| < r_{\min}$) are treated as if they were adjacent (since we remove short edges by collapsing them later, see Section 5.3.3). Hence, the same angle-criterion needs to hold for such edge-pairs.

For *non-adjacent* edges we need to ensure that no two arcs in A are closer than $r_{\min}$ to each other since then the sampling rate induced by the scale parameter $r_{\min}$ could no longer resolve both feature curves properly. We classify a pair of feature edges e_1 and e_2 as *potentially conflicting* if their minimum distance (edge-to-edge) is below $r_{\min}$. The triangulation between two potentially conflicting edges can essentially be one out of four possible configurations, which are depicted in Figure 4(b), (c), and (d). Configuration (b) can be excluded if $r_{\max}/r_{\min} < \sqrt{3}$ since then the minimum height of a triangle is larger than $r_{\min}/2$ and thus configuration (b) could be replaced by one of the configurations in (c). The required condition for non-adjacent edges is thus that at least one of the configurations (c) or (d) can be built with all edges having at least length $r_{\min}$, where in configuration (d) we additionally have to make sure that the angles β_1 and β_2 are both $\geq 2\alpha_{\min}$ allowing for two further triangles to be fitted inbetween. If we would only require $\beta_i \geq \alpha_{\min}$, this would reduce to configuration (c).

5. Feature Network Adaptation

The idea of our scale adaptation scheme is that we start with a given feature network N and then iteratively convert it into a network N' that is conforming to a given scale (see example in Figure 5). In order to make N conform to $r_{\min}$, we iterate a four step procedure as shown in Figure 2.

During the *proximity based conflict detection* phase, we first resample each arc of the network in order to satisfy the feature edge length condition (Section 5.2.1). Secondly, we identify conflicts in the current FCN (Section 5.2.2). In the *feature curve selection* phase we first compute a weight coefficient for each feature edge to rate its relevance (Section 5.3.1). In the third step we solve a global labeling problem that finds the maximum set of non-conflicting edges in the network (Section 5.3.2). Because these three steps cannot resolve complex constellations of multiple extraordinary feature vertices lying closer to another than $r_{\min}$, we have to perform a fourth step in which singularities are merged if required (Section 5.3.3). Since this changes the FCN topology, we have to run the entire four-step procedure again, starting with the resampling of the arcs. We do this until no further changes occur. This procedure always converges, since at least one edge is removed in each iteration. Usually, it converges after 2-5 iterations. The pseudo-code of the entire procedure is given in the supplemental material.

5.1. Initial Feature Network

We used three different feature detection methods to generate the initial FCNs N in our experiments: Variational Shape Approximation [CSAD04], the feature detection approach presented in [YBS05], and live-wire mesh segmentation [ZZCJ14]. The initial network has the same topology as the feature curves (e.g. segmentation boundaries as in Figure 5(a) or crest lines) and with arcs geometrically following relevant feature curves on the input surface.

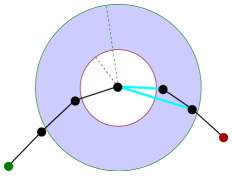
Note, that we preserve all features of the initial network that are not suppressed by a stronger feature. E.g. the VSA algorithm generates long streched patches in regions with constant curvature along one direction (e.g. inside of Rocker Arm, Figure 8). Since there are no stronger features nearby they are preserved in the abstraction.

5.2. Proximity Based Conflict Detection

Taking a feature network as input we have to determine whether it is scale conforming (cf. Section 4). In order to do this efficiently we have to resample the network. Non-conformity can then be represented as vertex and edge conflicts.

5.2.1. Feature Curve Resampling

Since the edge lengths along the arcs of the initial feature network N might not lie in the prescribed scale interval S, we apply a resampling procedure to each arc. Let $p_1, \ldots, p_n$ be a set of dense samples along an arc in the initial network. Our goal is to find the best possible piecewise linear approximation $v_1, \ldots, v_k$ such that the length of each resulting feature edge (v_i, v_{i+1}) lies in S. We formulate this optimization problem as a shortest path search.



We define a search graph as follows: $p_1, \ldots, p_n$ are the nodes of the graph (black points) and we define a directed edge (cyan) between p_i and p_j ($j > i$) if their Euclidean distance lies in the prescribed scale interval S (light blue). We assign a weight to this edge which is proportional to the geometric deviation (i.e. integral Euclidean distance) of the edge (p_i, p_j) from the (sub-)polygon $p_i, \ldots, p_j$. The optimal re-sampling is then found as the shortest path from p_1

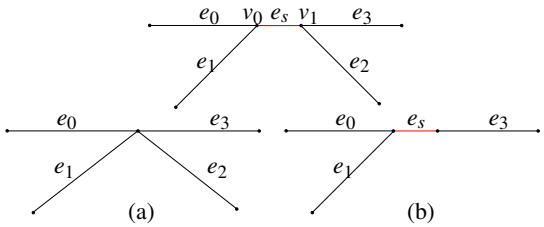


Figure 6: *Resolution of vertex conflicts: Vertex conflicts occur if $||v_0 - v_1|| \leq r_{\min}$ (top row). There are two possibilities to resolve such a conflict. (a) The vertices are collapsed into a common vertex. (b) The incident short edge is integrated into a neighboring arc. The choice between (a) and (b) is made by the global optimization (cf. Section 5.3.2). If the binary value of the edge e_s results in 0, (a) is applied. Otherwise, if it is 1, one of the adjacent vertices will have been downgraded to a regular vertex by the optimization.*

(green) to p_n (red) by using Dijkstra's algorithm. We label the nodes that are visited on this shortest path as $v_1, \ldots, v_k$. In case of loops where $p_1 = p_n$ we run Dijkstra's algorithm several times from different starting points to find the best p_1.

In the following two situations we cannot resample the feature curve to have all lengths in S. An obvious problem occurs when the entire initial arc is shorter than $r_{\min}$. This is handled by assigning a special *short* edge e_s with $||e_s|| < r_{\min}$ to this arc. Short edges will be removed in the post-processing step (Section 5.3.3) by collapsing them. Another, less obvious problem occurs if the arc length is larger than $r_{\max}$ but shorter than $2\,r_{\min}$ or more generally in an interval of the form $[k \cdot r_{\max}, (k+1) \cdot r_{\min}]$, which does not contain a solution in the given bounds. In this case we assign a special *long* edge e_l to the beginning or end of this arc with $||e_l|| > r_{\max}$.

5.2.2. Edge and Vertex Conflict Detection

In the resampled FCN it is possible to identify edge- and vertex-pairs that violate the conditions which need to be fulfilled in a scale conforming network defined in Section 4.

Edge-Conflicts: Neighboring edges that do not fulfill the angle criteria are labeled as conflicting. Also, the unconnected potentially conflicting edges described in Section 4 are tested for the listed configurations. If none of these configurations apply the edge-pair is labeled as conflicting.

Vertex-Conflicts: We define a pair of extraordinary vertices v_0 and v_1 as conflicting if and only if $||v_0 - v_1|| < r_{\min}$. If the FCN contains a short edge e_s (top row in Figure 6), it is incident to two extraordinary vertices $v_0, v_1 \in V^*$. In case the valence of these vertices is one, the short edge is not representable in the network and it will be removed. Otherwise, either v_0 and v_1 are collapsed into a common vertex (Figure 6a), or e_s is fused into one of the neighboring arcs. In the latter case, one of the two extraordinary vertices needs to be downgraded to a vertex with valence two (Figure 6b). e_s is then removed in the resampling step of the next iteration.

5.3. Feature Curve Selection

To achieve conformity for the FCN we have to resolve the detected conflicts found in Section 5.2. This is done by suppressing conflicting edges that are not as relevant as the edges they are in conflict with. Thus, we extract a globally optimal set of feature curves, not only respecting the amount of curves but also their quality. Finding such a globally optimal set outperforms possible greedy alternatives (e.g. farthest point/edge/curve sampling) since all conflicts and weights are regarded simultaneously, which is not possible for greedy solutions that inevitably lead to less appealing results.

5.3.1. Feature Edge Weights

As stated before, the relevance of a feature edge e can depend on several different properties (cf. Section 3). We account for both single-edge and combined-edge quality criteria by assigning weights to individual edges as well as to pairs of edges.

We compute individual edge weights as a product of different weighting factors:

$$w(e) = I(e) \cdot L(e) \cdot \mathrm{Sym}(e) \cdot \mathrm{Loop}(e).$$

The factor $I(e)$ measures the local sharpness by averaging the maximum (= across-feature) curvature $\kappa_{\max}$, computed using the shape operator [ACSD*03], along the segment boundary polygon $p_i, \ldots, p_j$ which is approximated by e (cf. Section 5.2.1):

$$I(e) = \frac{\sum_{k=i}^{j-1} \|p_{k+1} - p_k\| \left(|\kappa_{\max}(p_k)| + |\kappa_{\max}(p_{k+1})|\right)/2}{\sum_{k=i}^{j-1} \|p_{k+1} - p_k\|}.$$

In general, a feature edge is embedded in a connected feature curve and its length hints at the relevance of respective edges [DGGDV11]. Hence, the second factor $L(e)$ measures the length of the feature curve, to which e belongs. A simple choice would be to take the length of the respective arc in N. However, this would ignore the fact that feature curves can geometrically (and "semantically") extend beyond extraordinary vertices. The algorithms in [YBS05] and [ZZCJ14] provide us with an appropriate segment membership. In the case of VSA, we compute the length of a supporting feature curve for each feature edge. Starting with the arc to which e belongs, we extend this curve at both ends by adding that arc, which continues the existing feature curve in the most straight (i.e. tangent continuous) direction. The extension stops when the straightest arc no longer corresponds to a boundary of the same VSA patch as the starting arc. This stopping criterion restricts the extension to T-type-vertices in the feature network and makes sure that each feature curve remains a subset of a VSA segment boundary.

Symmetry is an important perceptual feature to identify a shape [MZL*09]. Hence, $\mathrm{Sym}(e)$ weights symmetric curve segments stronger than non-symmetric segments, i.e.:

$$\mathrm{Sym}(e) = \begin{cases} \alpha_{\mathrm{sym}} & \text{, if } e \text{ has symmetric edge } e' \\ 1 & \text{, otherwise} \end{cases}$$

where α_{sym} is a constant factor. We choose $\alpha_{\mathrm{sym}} = 2$ in our experiments. Symmetric feature curves are detected in a preprocessing step (e.g. with [MGP06]). Each edge that is labeled symmetric has a second (symmetric) representative, which only exists as long as its symmetric counterpart is preserved.

A feature curve that is connected to a closed loop describes a perceptual component, which should be preferred:

$$\text{Loop}(e) = \begin{cases} \alpha_{\text{loop}} & \text{, if } e \text{ lies on a loop} \\ 1 & \text{, otherwise.} \end{cases}$$

We choose $\alpha_{\text{loop}} = 100$ in our experiments.

The geometric quality of the result can be improved by not only considering the weight of individual feature edges but by also regarding pairs of edges. Smoothly connected segments convey the shape of an object better than highly curved segments and are considered more relevant [DGGDV11]. In order to promote connected curves over fragmented segments and interleaved edges approximating parallel curves, we introduce a smoothness measure $a_s(e_i, e_j)$. For this we consider all pairs of edges $e_i, e_j \in E$ which are adjacent to a common feature vertex. We choose:

$$a_s(e_i, e_j) = \begin{cases} \cos^p(2 \cdot \gamma_{i,j}) & \text{, for } \gamma_{i,j} \leq \frac{\pi}{4} \\ 0 & \text{, otherwise} \end{cases}$$

where $\gamma_{i,j}$ is the angle between e_i and e_j. $a_s(e_i, e_j)$ smoothly decreases from 1 to 0 as $\gamma_{i,j}$ increases from 0 to 45 degrees. The parameter p controls the fall-off. In all experiments we set $p = 2$.

For the purpose of quad-meshing, feature segments should be continued smoothly and intersect orthogonally to establish high quality meshes [BCE*13]. We can evaluate parallelism and orthogonality by introducing a combined weight $a_{||,\perp}$ between all edges that are within a prescribed radius:

$$a_{||,\perp}(e_i, e_j) = \cos^{2p}(2 \cdot \gamma_{i,j})$$

$a_{||,\perp}$ is close to 1 for angles near multiples of 90 degrees, which favors orthogonal configurations and parallel edge configurations.

5.3.2. Conflict Removal and Feature Edge Selection

In order to find the maximum conflict-free subset of the feature edges in N we formulate the problem-statement described in Sections 3 and 4 as a binary labeling problem for a set of optimization variables b_i, each indicating whether the corresponding feature edge $e_i \in E$ belongs to the conflict-free subset ($b_i = 1$) or not ($b_i = 0$). Since black-box numerical solvers can solve linear constrained problems more efficiently than quadratically constrained problems, we formulate the optimization as a linear program (e.g. by introducing binary pseudo variables).

A conflict-free subset of E is *optimal* if it maximizes the objective function:

$$\begin{aligned} T + \lambda_0 U_s + \lambda_1 U_{||,\perp} \\ T &= \sum_i b_i \cdot w(e_i). \\ U_x &= \sum_{i,j} a_{i,j} \cdot a_x(e_i, e_j). \end{aligned} \tag{1}$$

The objective function is composed of the terms T and U referring to the respective single-edge and combined-edge weights (cf. Section 5.3.1) with weighting factors λ_0 and λ_1 set by the user. In all our experiments we choose $\lambda_0 = 100$. For quad meshing related experiments we set $\lambda_1 = 10$, otherwise $\lambda_1 = 0$. For the combined objective, we introduce an additional set of binary pseudo variables $a_{i,j}$ which merely indicate if the combined term between e_i and e_j is active ($a_{i,j} = 1$) or not ($a_{i,j} = 0$). In case the combined term is not computed for the edges e_i and e_j (e.g. edges are not adjacent to a common vertex for the smoothness term) then $a_{i,j} = 0$. Otherwise the boundary constraints:

$$a_{i,j} \leq b_i \quad \& \quad a_{i,j} \leq b_j \tag{2}$$

make sure that the combined term is active only if both involved edges are.

In the following we will further refine this mathematical model to generate scale-conforming networks. For this we translate conflicts described in Section 5.2.2 into constraints of the optimization.

If the feature edges e_i and e_j have a conflict (according to the definition in Section 4) then we add a constraint of the type

$$b_i + b_j \leq 1 \tag{3}$$

to the optimization problem, which makes sure that not both edges can belong to the optimal subset. Note that symmetric edge pairs e_i and e_i' share the same optimization variable b_i, since a symmetric edge exists if and only if its symmetric counterpart exists. In case of n-fold symmetries n edges share the same optimization variable.

To handle vertex conflicts (cf. Section 5.2.2) where extraordinary feature vertices lie closer than $r_{\min}$ to each other, we introduce further binary variables c_i, which indicate for each extraordinary feature vertex $v_i \in V^*$ whether it should be kept ($c_i = 1$) or be removed ($c_i = 0$). First, for every pair of extraordinary feature vertices v_i and v_j whose distance is less than $r_{\min}$, we add a constraint

$$c_i + c_j \leq 1, \tag{4}$$

which is analog to the constraint for conflicting edges. Being labeled as "remove" does not necessarily mean that an extraordinary vertex is deleted. It can be sufficient to downgrade it to a regular feature vertex and then remove it in the feature arc resampling step of the next iteration, as described in Section 5.2.2. I.e. one of the conflicting vertices needs to become regular, which means its valence must become less or equal to 2. This condition for a vertex v_i can be formulated as another constraint:

$$(1 - c_i) \sum_{e_j \in \text{one-ring}(v_i)} b_j \leq 2 \tag{5}$$

A special case of this condition applies to extraordinary vertices v_i with valence 1 before conflict resolution. Here, the one adjacent edge e_j has to be removed as well, leading to the constraint:

$$(1 - c_i) \cdot b_j \leq 0. \tag{6}$$

These quadratic constraints can be linearized in the following way: For every edge e_i a pseudo variable p_i is added. An equivalent formulation to Constraint 5 (and analogously Constraint 6) is:

$$\begin{aligned} \sum_{e_j \in \text{one-ring}(v_i)} p_j &\leq 2 \\ p_j - b_j \leq c_i \quad \& \quad b_j - p_j &\leq c_i. \end{aligned} \tag{7}$$

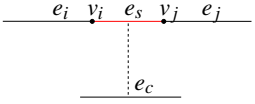
With the above constraints, we guarantee that all conflicts are properly handled. We can further improve the quality of the resulting FCN by regarding two further types of conflicts. So far we might end up with isolated short edges, which cannot be resampled and

are not representable at the given resolution. To avoid this we introduce another constraint for all edges $e_s = (v_i, v_j)$ shorter than $r_{\min}$:

$$\sum_{e_k \in \text{one-ring}(v_i) \setminus e_s} b_k + \sum_{e_k \in \text{one-ring}(v_j) \setminus e_s} b_k \geq b_{e_s}, \qquad (8)$$

which implies that if e_s is removed ($b_{e_s} = 0$) then the constraint does not have any effect, but if the short edge is kept ($b_{e_s} = 1$) then at least one of the two end vertices must be connected to at least one more edge, leaving no isolated short edges in the abstraction.

As described above (Section 5.2.2), we will remove a short edge e_s ($b_{e_s} = 0$) by collapsing its two end vertices (cf. Section 5.3.3). Then we must consider the following situation on the right: It can occur that there is a set E_c of other feature edges (e.g. e_c) that had a conflict with e_s. These conflicts (dashed line) are inherited after the collapse by the edges adjacent to e_s (e.g. e_i and e_j). Since these are foreseeable conflicts, we can take them into account preemptively by adding another set of constraints to our optimization problem. For every short edge $e_s = (v_i, v_j)$ and every edge e_c from its conflicting set E_c we require



$$b_c + b_i + b_j \leq 2 \qquad (9)$$

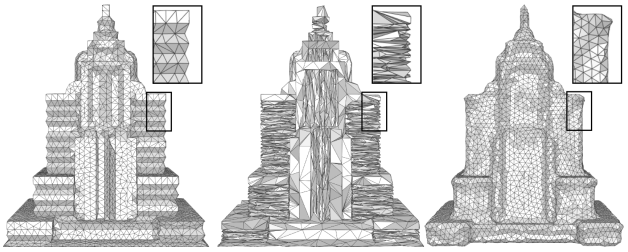
for all pairs of edges e_i and e_j that are adjacent to v_i and v_j respectively. This constraint can be interpreted as follows: if e_s wins against all its conflicting edges (i.e. $b_s = 1$) then $b_c = 0$ (according to Constraint (3)) for all edges in E_c and thus Constraint 9 is automatically satisfied. If e_s loses ($b_s = 0$) and an edge $e_c \in E_c$ is preserved by the optimization ($b_c = 1$) then in each pair e_i, e_j only one edge can be kept.

Overall, the task of finding the optimal non-conflicting subset of the given feature network N amounts to solving a linear program (1) with linear constraints (3)-(4), and (7)-(9). These types of optimization problems can be solved quite efficiently by off-the-shelf solvers such as the linearly constrained mixed integer solver by GUROBI [GO15].

The objective of this optimization is the preservation of as much feature information as possible. Since the edges are treated separately it can occur that feature curves are fragmented. Applications such as segmentation layout generation might require the retainment of a patch structure. Incorporating higher order structure and treating an entire feature curve as a whole (and thus avoiding fragmentation and invalid segmentation layouts) can be implemented by assigning the same binary optimization variable to all edges belonging to a curve.

5.3.3. Short Edge Collapsing

The result of the labeling problem is a set of binary variables that indicate, which edges belong to the maximum non-conflicting subset. Short edges e_s that are assigned $b_s = 0$ are now removed from the network by edge collapses where we pull the vertex with lower sharpness value into the one with higher sharpness. This process together with the deletion of suppressed regular edges leads to topological changes in the feature network, since entire branches can disappear and formerly separate arcs can be joined. Hence, we iterate the process by feeding the result back into the resampling step (cf. Section 5.2.1).



our method	Quadric Decimation	Isotropic Remeshing
0.0195	0.0194	0.0883

Figure 7: *Comparison with Quadric and Isotropic Remeshing. The Quadric decimation was performed until the mesh-complexity (vertices) was equal, while isotropic remeshing was performed for the same target edge length. The numbers below give the Hausdorff distance (Hd) (bounding box diagonal of length* 1.4*). Although the Hd of the Quadric decimation is similar to that of our abstraction, ours has well-shaped triangles and suppresses features that cannot be represented, while the decimated object has many degenerate, pointy triangles aligning along the features. The isotropic remeshing has a much higher Hd, since features are smoothed away.*

6. Results and Applications

Figure 8 shows a variety of exemplary FCNs generated by our method. We observe that for each scale interval, continuous and prominent feature-lines are preserved, while less salient curves, which are not representable are suppressed. Exact timings and measurements of the optimization are given in the supplemental material. As proof of concept we will present two different applications, which demonstrate the usefulness of the generated FCNs.

6.1. Isotropic Remeshing

Advances in triangle remeshing focus on goals such as high quality, feature preservation, or fidelity [ADVDI03, AUGA08]. While features can be preserved, it is not possible to suppress them in a controlled manner for a given target edge length. For demonstration, we apply quadric decimation [GH97] and isotropic remeshing [BK04] on the Skyscraper model in Figure 7 (middle and right). We can observe that quadric decimation, preserves features although they cannot be represented properly at the given resolution, which leads to many degenerated and pointy triangles. Isotropic remeshing on the other hand smoothes the features away. Both methods rely on local shape information. In contrast, our feature curve suppression method takes the global feature spacing into account and generates a subsampling of features which leads to well-shaped triangles, which align along feature curves (Figure 7(left)).

To achieve this we leverage the FCN as input to the isotropic remeshing algorithm described in [BK04]. Figures 1 and 8 show the scale adapted feature curves and the respective abstracted triangle meshes at different target scales S (fine-to-coarse). In contrast

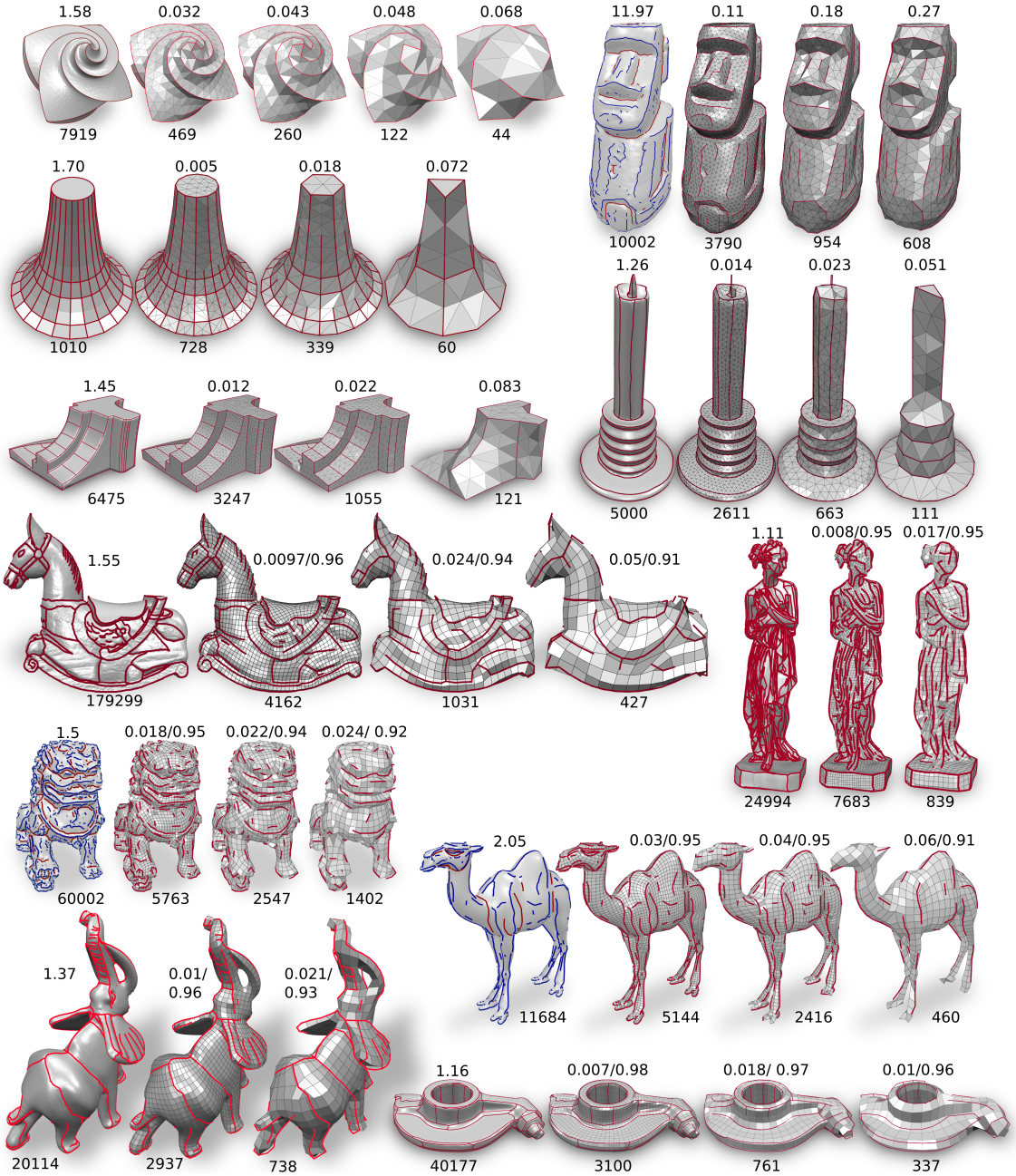


Figure 8: *Scale conforming FCNs at different scales. Initial feature sets are depicted leftmost with the length of the bounding box diagonal given above the respective object. These were generated with VSA (Trumpet, Octaflower, Fandisc, Iphigenie, Rockerarm), [YBS05] (Moai, Chiniese Dragon, Camel), or Livewire (Candel, Isidore Horse, Elephant). Extracted FCNs at different scales were used as input to isotropic triangle remeshing [BK04], or level of detail quad meshing [ECBK14]. The number below the objects gives the mesh complexity number of vertices). The one above gives the Hausdorff distance (Hd) for triangle meshes and Hd/average Scaled Jacobian (aSJ) for quad meshes.*

to previous feature suppression mechanisms, where curves are removed via filtering or smoothing, we can observe a subsampling of the feature-curves which is based on global scale parameter.

We compare the curvature-filtering approach presented in [YBS05] to our feature subsampling in Figure 9. Examples are generated with the curvature thresholding parameter set to $T = 0$, $T = 0.5$ and $T = 0.8$. In the case where no thresholding is performed ($T = 0$) we observe several small and dense features. Triangles that align along these features do not satisfy the target resolution. We increase T to remove small scale details. At the same time features are removed and smoothed away in the remeshing procedure which could have been represented at the given scale

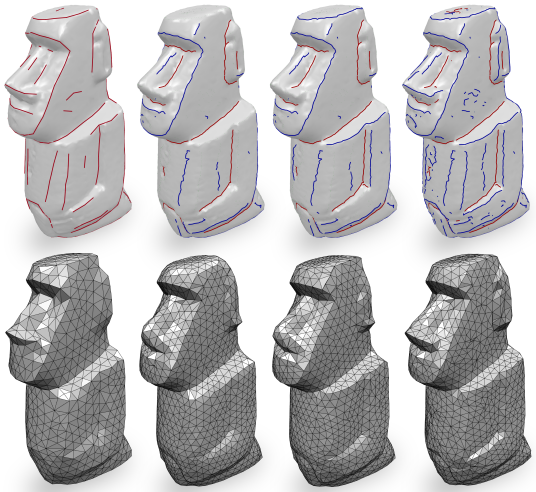
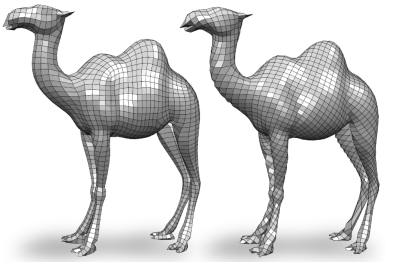


Figure 9: *Comparison of our feature line abstraction method to curvature thresholding. The top row depicts the boundary constraints used for isotropic triangle remeshings (bottom row). The numbers below give Hausdorff distance (Hd)/vertices. The left column shows results from our feature line abstraction. On the right, isotropic remeshing constrained by the ridges and ravines from [YBS05] is applied with decreasing curvature thresholding parameter T. For $T = 0$ the prescribed resolution cannot be guaranteed and shows a much higher mesh complexity, while for increasing T features are removed that could have been represented (e.g. tip of the nose, ear). It is not surprising that the Hd is smaller for $T = 0$, since all features are preserved during remeshing.*

(e.g. the tip of the nose or the ears in Figure 9). With our method, the feature curves are adapted to the target resolution, i.e., all features which can be represented at the given scale are preserved and resulting triangles are well shaped. Further comparisons are given in the supplemental material.

6.2. Guiding Constraints for Level-of-Detail Quad Meshing

State of the art quad meshing and quad layout methods are not able to suppress or select features based on their density. They create high quality surface representations only if they are provided with (or precompute) proper guiding constraints. These guiding constraints give a direction along which the quads should align. E.g. the image below (two camels) depicts two quad meshes (generated with [EBCK13, ECBK14]). On the left, our feature abstractions were supplied as guiding constraints, while for the right mesh no constraints were given. Note that in contrast to the image on the right, the quads in the left image align along features



quad meshing with **our method** as guiding constraints | without guiding constraints

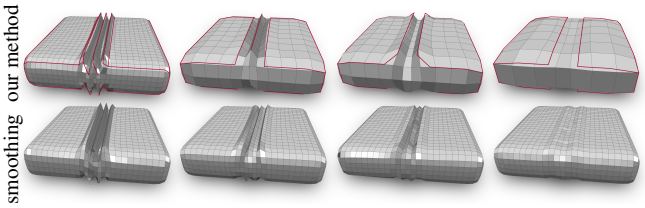


Figure 10: *Comparison of our FCN adaptation to conventional filter approaches in the context of quad mesh generation. The top row shows our extracted FCNs used as guiding constraints for Level-of-Detail Quad Meshing [ECBK14] with the respective target edge length. The bottom row shows the results obtained by applying a smoothing filter to the input mesh to suppress fine detail before quad meshing. Here, the target edge length is not adapted to avoid undersampling artifacts. Notice that in the top row, the number of ribs is incrementally decreasing while in the bottom row, the number of ribs remains constant and only their amplitude decreases.*

and curvature directions, which demonstrates the need for such guiding constraints. So far, guiding constraints are usually either manually supplied or obtained from filtered curvature directions [KNP07, BZK09, NPPZ12, CBK12, BCE*13, ECBK14]. Although user-supplied constraints lead to high quality results, their acquisition is tedious. Guiding constraints obtained via filtering are sensitive to noise and ignorant of the feature density. Furthermore, certain thresholds need to be set manually by a user. In Figure 10, we demonstrate the effect of feature suppression by smoothing (bottom row) in contrast to our sub-sampling approach (top). For comparison we generate quad meshes guided by curvature constraints, which were smoothed in advance to remove small-scale details. Note that the number of ribs remains constant and only their amplitude decreases, while the number of ribs is incrementally reduced with our sub-sampling approach and hence gives a representation of the feature set at different resolutions.

Furthermore, we compare quad meshes generated with curvature thresholded guiding constraints [BZK09] to those constrained by our scale conforming FCNs in Figure 11. In Figure 11a we choose a high curvature threshold, to preserve only strong features. By increasing the threshold further (11b) nearly all constraints are removed simultaneously since all features of the same magnitude will either be preserved or removed. Also, less prominent features (e.g. the collar around the Chinese Dragons neck or the eyes' region in Figure 11a) are not represented in the curvature thresholded feature set and thus smoothed away. They can only be included by thresholding if all other features of the same strength are incorporated. This can lead to overconstrained parametrizations. Our subsampling preserves the representable features and quads align along them with high element quality (the average scaled Jacobian is close to that of the nearly unconstrained mesh). Hence, using the scale conforming FCNs obtained from our method as soft guiding constraints introduces a high-quality, automatic alternative in this context. By combining our scale conforming approach with Level-of-Detail Quad Meshing from [ECBK14] we can suppress densely spaced features and noise, while preserving dominant large-scale features. The final quad meshes are extracted with QEx [EBCK13]. Figure 8 shows a variety of quad meshes at different levels of de-

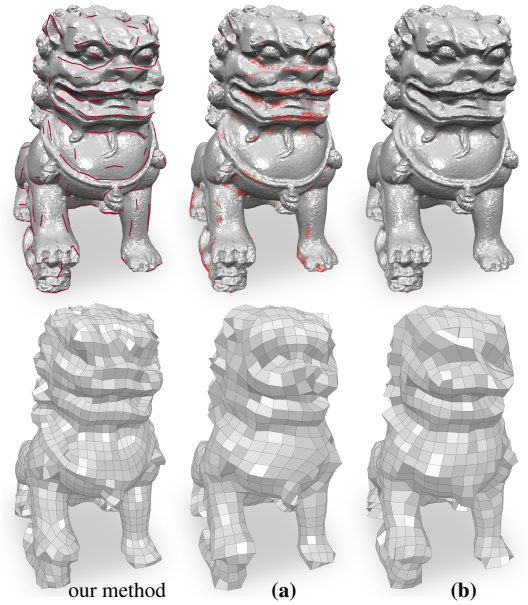
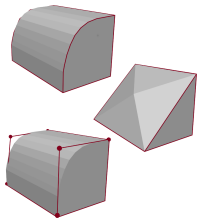


Figure 11: *Comparison of our scale conforming FCN as guiding contraints for quadmeshing to contraints induced by setting curvature thresholds as in [BZK09]. The top row depicts the guiding constraints. We choose a high curvature threshold in (a) to preserve only strong features and avoid degeneracies in the parametrization. If this threshold is increased in (b) nearly all strong features are removed simultaneously. The numbers below give the Hd/average Scaled Jacobian (aSJ). While in (a) the quads align along the features the aSJ is much lower than in (b), which has a high aSJ due to the low number of constraints, but quads do not align well along features. We can observe that the aSJ is similar to that of (b), while still aligning well to the features at the given scale.*

tail, which were generated using the extracted FCNs as soft guiding constraints.

Limitations Since we strive to maintain a maximum amount of salient feature curves in the final network, we cannot guarantee that it will result in a connected B-Rep, as it is done in [MZL*09]. Also, the connectivity of the final feature-graph is strongly dependent on the initial network. E.g. the feature curves generated with [YBS05] are mostly isolated ridges or ravines without many junctions. The gaps between curves are also visible in our scale conforming FCNs, which should be acknowledged when choosing the method to generate the initial feature curves. Moreover, we do not close gaps between feature lines during the abstraction for two reasons. First of all, we do not want to "invent" additional information to the initial feature set, since we would require a heuristic which selects two curves to close the gap. Secondly, we can guarantee that the algorithm terminates since we remove at least one feature edge in each iteration. Extending features could affect termination.

Furthermore, our arc resampling strategy restricts the possible abstractions. We only resample along the arc and do not take any points into account that deviate from the arc. E.g. the image to the right shows a rounded surface. Our resampling cuts this curve off (middle). A better abstraction (in the sense of a least squares error) might be to extend the curves as it is done in the image below. The benefit of our approach is that it has a comparatively small solution space, allowing an efficient computation. To extend features as discussed above would involve finding an appropriate formulation to solve this problem efficiently.



Another limitation derives from the NP-hardness of binary optimization. This can affect performance in cases with a large amount of inter-dependent conflicts. A greedy approximation strategy could be of value in such a setting. However, in our examples the duration of the optimization procedure ranged from 0.0023s (Octaflower) to 72.09s (Iphigenie) with an Intel Core i7-4770 CPU. More details on timings are given in the supplemental material.

7. Conclusion

We have presented a computational approach to the intuitive notion of scale conforming FCNs, which poses it as a binary labeling problem. It guarantees that the spacing between feature curves is large enough such that they can be meshed properly at a prescribed resolution.Conflict suppression was implemented as a global optimization procedure, allowing for all detected conflicts to be handled at once, without the need for local or greedy decisions. The resulting feature line based abstractions were used to generate feature preserving level-of-detail mesh approximations.

Tracing the arcs for scale conforming FCNs relies on the assumption that smoothly connected edges define a perceptual segment. We could refine this heuristic by considering geodesic curvature only. Moreover, our approach provides the option to take additional information into account, which can be provided by a user. Also, including further semantic information, e.g. adding texture awareness to the abstraction process, could be worth investigating.

A further interesting direction of research could be to make the FCN conform to a scalar sizing field over the mesh, which could e.g. be computed from the local feature size or given by a user. This would allow to resolve different levels of detail on one object, which could be used to e.g. remove noise from a feature network.

The scale conforming feature curves computed by our algorithm provide a new automatic alternative for feature based level-of-detail methods. Instead of requiring user supplied feature constraints we provide the means to automatically generate feature abstractions at variable levels of detail. As input a user has to precompute the initial FCNs. However, this is not a problem since features can be quite dense, because small, weak features are suppressed by the algorithm. We believe that the detection and preservation of salient and representable features can be useful for various kinds of level-of-detail algorithms.

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Union's Seventh

Framework Programme (FP7/2007-2013)/ERC grant agreement n° [340884]. Several models are provided courtesy of INRIA and Yutaka Ohtake by the AIM@SHAPE-VISIONAIR Shape Repository. Furthermore, we thank Hans-Christian Ebke for providing us with the software for Level-of-Detail Quad Meshing and QEx. Last but not least, we would like to thank Jan Möbius for creating and maintaining the geometry processing framework OpenFlipper as well as the reviewers for their insightful comments.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, ACM, pp. 485–493. 6

[ADVDI03] ALLIEZ P., DE VERDIRE E., DEVILLERS O., ISENBURG M.: Isotropic surface remeshing. In *Shape Modeling International, 2003* (2003), IEEE, pp. 49–58. 1, 8

[AUGA08] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring, Mathematics and Visualization* (2008), Springer. 8

[BCE*13] BOMMES D., CAMPEN M., EBKE H.-C., ALLIEZ P., KOBBELT L.: Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4 (July 2013), 98:1–98:12. doi:10.1145/2461912.2462014. 7, 10

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (New York, NY, USA, 2004), SGP '04, ACM, pp. 185–192. doi:10.1145/1057432.1057457. 1, 8, 9

[BPK98] BELYAEV A. G., PASKO A. A., KUNII T. L.: Ridges and ravines on implicit surfaces. In *Computer Graphics International, 1998. Proceedings* (1998), IEEE, pp. 530–535. 2

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 77:1–77:10. doi:10.1145/1576246.1531383. 1, 10, 11

[CBK12] CAMPEN M., BOMMES D., KOBBELT L.: Dual loops meshing: Quality quad layouts on manifolds. *ACM Trans. Graph. 31*, 4 (July 2012), 110:1–110:11. doi:10.1145/2185520.2185606. 10

[CJL11] CHIANG C.-H., JONG B.-S., LIN T.-W.: A robust feature-preserving semi-regular remeshing method for triangular meshes. *The Visual Computer 27*, 9 (2011), 811–825. doi:10.1007/s00371-011-0555-1. 1

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 905–914. doi:10.1145/1186562.1015817. 2, 5

[CYW14] CAO Y., YAN D.-M., WONKA P.: Patch layout generation by detecting feature networks. *Computers & Graphics* (2014). 2

[Dem09] DEMARSIN K.: *Extraction of Closed Feature Lines from Point Clouds Based on Graph Theory*. PhD thesis, Numerical Analysis and Applied Mathematics Section, Department of Computer Science, Faculty of Engineering Science, Jan. 2009. URL: https://lirias.kuleuven.be/handle/1979/2027. 2

[DGGDV11] DE GOES F., GOLDENSTEIN S., DESBRUN M., VELHO L.: Exoskeleton: Curve network abstraction for 3d shapes. *Comput. Graph. 35*, 1 (Feb. 2011), 112–121. doi:10.1016/j.cag.2010.11.012. 2, 6, 7

[EBCK13] EBKE H.-C., BOMMES D., CAMPEN M., KOBBELT L.: QEx: Robust quad mesh extraction. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 168:1–168:10. doi:10.1145/2508363.2508372. 10

[ECBK14] EBKE H.-C., CAMPEN M., BOMMES D., KOBBELT L.: Level-of-detail quad meshing. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 184:1–184:11. doi:10.1145/2661229.2661240. 1, 9, 10

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 209–216. doi:10.1145/258734.258849. 8

[GO15] GUROBI OPTIMIZATION I.: Gurobi optimizer reference manual, 2015. URL: http://www.gurobi.com. 8

[KNP07] KÄLBERER F., NIESER M., POLTHIER K.: Quadcover-surface parameterization using branched coverings. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 375–384. 10

[LZH*07] LAI Y.-K., ZHOU Q.-Y., HU S.-M., WALLNER J., POTTMANN H.: Robust feature classification and editing. *IEEE Trans. Visualization and Computer Graphics* (2007). 2

[LZHM06] LAI Y.-K., ZHOU Q.-Y., HU S.-M., MARTIN R. R.: Feature sensitive mesh segmentation. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2006), SPM '06, ACM, pp. 17–25. doi:10.1145/1128888.1128891. 2

[MDS09] MI X., DECARLO D., STONE M.: Abstraction of 2d shapes in terms of parts. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2009), NPAR '09, ACM, pp. 15–24. doi:10.1145/1572614.1572617. 2

[MGP06] MITRA N. J., GUIBAS L., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH) 25*, 3 (2006), 560–568. 6

[MZL*09] MEHRA R., ZHOU Q., LONG J., SHEFFER A., GOOCH A., MITRA N. J.: Abstraction of man-made shapes". *ACM Transactions on Graphics" 28*, 5 (2009), #137, 1–10. 2, 6, 11

[NPPZ12] NIESER M., PALACIOS J., POLTHIER K., ZHANG E.: Hexagonal global parameterization of arbitrary surfaces. *Visualization and Computer Graphics, IEEE Transactions on 18*, 6 (2012), 865–878. 10

[NSP10] NIESER M., SCHULZ C., POLTHIER K.: Patch layout from feature graphs. *Comput. Aided Des. 42*, 3 (Mar. 2010), 213–220. doi:10.1016/j.cad.2009.11.002. 2

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 609–612. doi:10.1145/1015706.1015768. 2

[Rup95] RUPPERT J.: A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms 18*, 3 (1995), 548–585. 3

[SD08] SALA P., DICKINSON S.: Model-based perceptual grouping and shape abstraction. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on* (June 2008), pp. 1–8. doi:10.1109/CVPRW.2008.4562979. 2

[WB01] WATANABE K., BELYAEV A. G.: Detection of salient curvature features on polygonal surfaces. In *Computer Graphics Forum* (2001), vol. 20, Wiley Online Library, pp. 385–392. 2

[WG09] WEINKAUF T., GÜNTHER D.: Separatrix persistence: Extraction of salient edges on surfaces using topological methods. In *Proceedings of the Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2009), SGP '09, Eurographics Association, pp. 1519–1528. URL: http://dl.acm.org/citation.cfm?id=1735603.1735638. 2

[YBS05] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: Fast and robust detection of crest lines on meshes. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2005), SPM '05, ACM, pp. 227–232. doi:10.1145/1060244.1060270. 2, 5, 6, 9, 10, 11

[ZZCJ14] ZHUANG Y., ZOU M., CARR N., JU T.: Anisotropic Geodesics for Live-wire Mesh Segmentation. *Computer Graphics Forum 33*, 7 (2014), 111–120. doi:10.1111/cgf.12479. 2, 5, 6