The Visual Computer manuscript No.
(will be inserted by the editor)

Feature Sensitive Multiscale Editing on Surfaces

U. Clarenz[1], **M. Griebel**[2], **M. Rumpf**[1], **M. A. Schweitzer**[2], **A. Telea**[3]

[1] Institut für Angewandte Mathematik, Bonn University, Wegelerstrasse 6, 53115 Bonn, Germany
[2] Fachbereik Mathematik, Duisburg University, Lotharstrasse 65, 47048 Duisburg, Germany
[3] Eindhoven University of Technology, Department of Mathematics and Computer Science, Den Dolech 2, 5600 MB Eindhoven, Netherlands

Received: date / Revised version: date

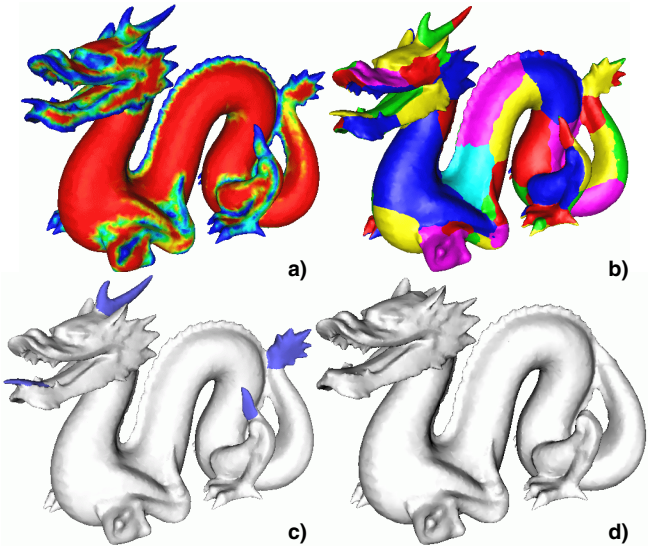

Fig. 1 Surface processing example: (a) feature detector (b) feature-guided surface decomposition (c) features selected for deletion (d) surface after feature deletion

Abstract A novel editing method for large triangular meshes is presented. It is based on a stable local surface classification and feature detection algorithm, the definition of a finite element matrix encoding a weighted coupling of adjacent mesh nodes, and an algebraic multigrid (AMG) algorithm. In particular, edges and corners are regarded as features on the surface. We detect features using an analysis of local zero and first surface moments, as computing these quantities is robust and noise resistant. The feature detection is encoded in a finite element matrix, passed to the AMG algorithm. The AMG algorithm generates a matrix hierarchy ranging from fine to coarse respresentations of the initial fine grid matrix. This hierarchy comes along with a corresponding multiscale of basis functions, which reflect the surface features on all hierarchy levels. We consider either these basis functions or distinct sets from an induced multiscale domain decomposition as handles for surface manipulation. Finally, we present a multiscale editor which enables boolean operations on this hierarchical domain decomposition and simple algebraic operations on the basis functions. Users can thus interactively design their favorite surface handles by simple grouping operations on the multiscale of the feature-sentitive basis functions or domains. Several applications on large meshes underline the effectiveness and flexibility of the presented tool.

Key words surface processing – algebraic multigrid – multiscale feature detection

1 Introduction

Flexible, interactive surface modeling is a challenging topic in computer graphics. In particular, multiresolution strategies have proved to be an efficient way for processing large triangular surface meshes [18, 20, 22]. Surfaces of a complicated shape and non trivial topology have to be treated and processed in an intuitive and interactive way [34]. Hereby, surface features such as edges and corners are of particular interest. The set of all surface features is usually characterized by different scales. Usually, one finds prominent, sharp, and long edges, together with less pronounced, slighty curved features, confined in smaller surface regions. Usually, such features separate the surface in a number of smooth regions that correspond intuitively to different object parts. Just as the edge features, these parts come at different scales, e.g. the dragon's horn and tongue on finer scales, and the head and body, on coarser scales. To our knowledge, this multiscale nature of surface features has not been considered so far. In this paper we present a novel approach to surface modeling which

- robustly detects features on large and small scales,
- computes a multiscale library of surface handles reflecting features, and
- enables a flexible interactive, and reliable multiscale surface editing.

In the following, we outline the main steps of the proposed method (see also Fig. 2 and the example in Fig. 1). The method

is based on a local zero and first moment analysis to classify features on discrete surface. The zero and first moment integral quantities are stable to compute and they give less noisy results compared to discrete curvature quantities. The resulting local surface classification, computed at the triangulation vertices of the surface, is encoded in a finite element stiffness matrix. Thereby, the matrix describes the coupling of regions on the surface. By construction, this coupling is much weaker along feature edges than in smooth areas. Next, an algebraic multigrid (**AMG**) method is applied to this matrix. The AMG delivers a matrix representation on multiple scales and an accompanying multiscale library of discrete basis functions, which can be seen as feature sensitive surface handles. In other words, the AMG delivers a multiscale representation of our surface classifier. Coarse levels show the main surface characteristics, i.e. the smooth regions separated by the most salient surface features. Finer levels show the (usually smaller) regions separated by less pronounced, detail surface features. To build general surface handles, an editor tool is presented which allows combining basis functions from the multiscale library. Figure 1 shows the different ingredients of the approach: the robust feature detection showing the weak coupling along feature edges, the AMG-based domain decomposition on a particular scale, several surface handles selected form the multiscale library, and finally the surface edited by deleting the selected handles.

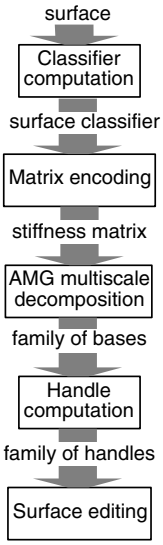

Fig. 2 Steps of the multiscale surface editing method

Review of related work The method we present here is related to other applications of AMG which are also related to preconditioning. In particular, in [28], AMG is used to segment images via a multiscale method. In these applications the coarsening is comparable to a hierarchical watershed algorithm [24], where homogeous regions on surfaces, bounded by curvature features, are extracted. Furthermore, AMG has been applied to optimal graph drawing applications [33]. Here, AMG serves again as an appropriate clustering algorithm. The common multiscale characteristic distinguishes these approaches in particular from other surface decomposition methods, such as those given in [35], where another watershed approach is taken into account, and in [10], where a combinatorical approach is presented.

One of the building blocks of our method is a reliable surface feature detection, an indispensable tool in image and surface processing. Features such as edges and corners in images have to be classified in a stable way to enable edge preserving image denoising [26, 1] and robust segmentation of image subdomains bounded by edges [8]. In image processing, a straightforward identification of edges can be based on an evaluation of the image gradient. A sufficiently large gradient is supposed to indicate an edge. Alternatively, a frequently considered edge indicator is the Canny edge indicator, which searches for extrema of the second derivatives in the gradient direction [13]. Furthermore, the structure tensor [31]) enables a robust classification of edges and edge direction in images.

Stable local classification of triangular meshes has been considered in surface applications too [19, 23] with the aim to improve surface processing. Feature detection is usually based on the measurement of dihedral angles [22] or on a local curvature analysis [19, 22]. An edge is supposed to be indicated by one sufficiently large principle curvature and the corresponding principle curvature direction is perpendicular to the edge on the surface. A well known approach for curvature evaluation on discrete surfaces is algorithm proposed in [25]. In [11] principal curvatures are evaluated based on a local projection of the mesh onto quadratic polynomial graphs. If concerned with large triangular and irregular grids, e. g. generated by marching cubes, such detectors are tedious to treat and a robust classification is hard to achieve. In critical applications features are usually extracted manually [17]. Various applications rely on a robust feature detection. In surface fairing a given initial, noisy meshes have to be smoothed, while simultaneously preserving edges on the surface [14, 11]. In recent mesh decimation tools, surface meshes are simplified while edge features are retained [32]. As a final application, we mention automatic texture generation, where it is desirable that the texture map is bounded by feature lines [23].

Moment analysis for feature detection has already been present in the graphics and computer vision areas [29, 21]. Here we focus on using moments as a multiscale feature classification tool and provide details for their robust computation. Finally, there is a wealth of literature addressing the topic of surface editing, such as [34] and [2]. However, to our knowledge, no similar methods based on *algebraic* multigrid (AMG) exist. Since the main message of this paper is the novel introduction of the AMG in the field of multiscale surface processing, we shall not insist on reviewing specific surface editing methods and tools.

The paper is organized as follows. In Section 2 we briefly review algebraic multigrid methods. Then the local classification of surfaces based on moments is discussed in Section 3. We will use this classification to define a matrix encoding the features of the surface in Section 4 and in Section 5 a mul-

tiscale library of surface handles will be computed applying algebraic multigrid to this matrix. The multiscale surface editor will be introduced in Section 6 and in Section 7 we present some applications. Finally in Section 8 we draw conclusions and indicate future work directions.

Notation

Before we develop our approach to multiresolution modeling, let us first briefly introduce some basic notations. For a detailed introduction to geometry and differential calculus we refer to [15]. Let us consider a closed and orientable surface $\mathcal{M} \subset \mathbb{R}^3$. Let $x : \Omega \to \mathcal{M} ; \xi \mapsto x(\xi)$ be some coordinate map from an atlas. For each point x on $\mathcal{M}$ the tangent space $\mathcal{T}_x\mathcal{M}$ is spanned by the basis $\{\frac{\partial x}{\partial \xi_1}, \frac{\partial x}{\partial \xi_2}\}$. By $\mathcal{T}\mathcal{M}$ we denote the tangent bundle. Measuring length on $\mathcal{M}$ requires the definition of a metric $g(\cdot,\cdot) : \mathcal{T}_x\mathcal{M} \times \mathcal{T}_x\mathcal{M} \to \mathbb{R}$. As the corresponding matrix notation we obtain the first fundamental form $g = (\mathrm{g}_{ij})_{ij}$ with $\mathrm{g}_{ij} = \frac{\partial x}{\partial \xi_i} \cdot \frac{\partial x}{\partial \xi_j}$, where $\cdot$ indicates the scalar product in $\mathbb{R}^3$. The inverse of g is denoted by $\mathrm{g}^{-1} = (\mathrm{g}^{ij})_{ij}$. The gradient $\nabla_{\mathcal{M}} f$ of a function f is defined as the representation of $\mathrm{d}f$ with respect to the metric g. In coordinates we obtain

$$\nabla_{\mathcal{M}} f := \sum_{i,j} \mathrm{g}^{ij} \frac{\partial (f \circ x)}{\partial \xi_j} \frac{\partial}{\partial \xi_i} .$$

We define the divergence $\mathrm{div}_{\mathcal{M}} v$ of a vector field $v \in \mathcal{T}\mathcal{M}$ as the dual operator of the gradient with respect to the L^2-product on $\mathcal{M}$ and obtain in coordinates

$$\mathrm{div}_{\mathcal{M}} v := \sum_i \frac{\partial}{\partial \xi_i} ((v_i \circ x)\sqrt{\det \mathrm{g}}) \frac{1}{\sqrt{\det \mathrm{g}}} .$$

Finally, the Laplace Beltrami operator $\Delta_{\mathcal{M}}$ is given by

$$\Delta_{\mathcal{M}} u := \mathrm{div}_{\mathcal{M}} \nabla_{\mathcal{M}} u .$$

Let us denote by N the normal field on the surface $\mathcal{M}$.

2 A brief introduction to AMG

In this section we give a short review of the basic algebraic multigrid algorithm (for scalar PDEs) and the heuristics which led to its development, see [30] for a detailed introduction to AMG.

Algebraic multigrid methods were first introduced in the early 1980's [3–6, 27] for the solution of linear systems $Au = f$ coming from the discretization of scalar elliptic PDEs. The development of AMG was led by the idea to mimic (geometric) multigrid methods, i.e. their functionality and convergence behavior, in applications where a hierarchy of (nested) meshes and interlevel transfer operators could not (or only with huge effort) be provided. The amount of input information for the iteration scheme should be minimal, i.e., the linear system itself should provide all the information needed for the algorithm.

Roughly speaking, we define a sequence of matrices A^l from the input (fine level) matrix $A^0 := A$ via the a natural coarsening (often named Galerkin projection)

$$A^l := R^l A^{l-1} P^l = (P^l)^T A^{l-1} P^l ,$$

where P^l is an appropriately chosen prolongation matrix (encoding how coarse scale (l) basis functions are combined using the basis functions on the finer scale ($l-1$). In particular, AMG constructs a sequence of appropriate prolongation matrixes $\{P^l\}_{l=0,\dots,L}$ using information from the matrix A^{l-1} on the previous level $l-1$ only. The construction of a prolongation matrix can also be viewed as the construction of a problem-dependent basis $\{\Psi^{l,i}\}$. We construct a coarser basis $\{\Psi^{l,i}\}$ which captures the appropriate features relevant for the approximation of the corresponding continuous problem, i.e. the underlying differential operator (cf. Section 4). The theory and the design of efficient AMG packages is rather involved. We here require the basic AMG capabilities. There are several suitable AMG packages available on the Web (e.g. under *www.mgnet.org*, see also the software discussed in [33]). Let us recall the essential ingredients of AMG algorithms. In

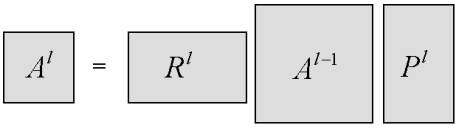

Fig. 3 Coarse matrix A^1 definition illustrating matrix sizes

general, any AMG implementation works as follows (see also Fig. 4):

- Given fine grid matrix $A^0 := A$.
- Construct prolongation P^1; i.e. coarse basis functions $\{\Psi^{1,i}\}$.
- Define restriction $R^1 := (P^1)^T$.
- Define the coarse matrix $A^1 := R^1 A^0 P^1$ via the Galerkin identity.
- Recursive application gives a sequence of prolongation P^l and restriction R^l matrices, as well as matrices A^l on all levels $l = 0, \dots, L$.

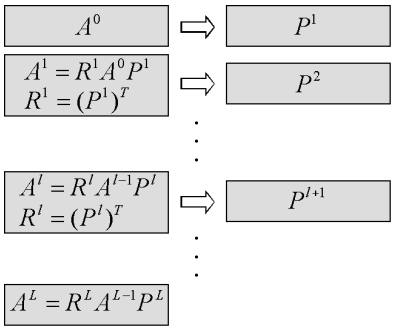

Fig. 4 General AMG construction

The fundamental ingredient in this AMG construction is the notion of *algebraic smoothness*. With the help of such a smoothness measure we can set up a reduced graph of the matrix from

which we can then "merge" fine level basis functions $\Psi^{l-1,i}$ on level $l-1$ in an appropriate fashion to define the coarse basis $\{\Psi^{l,i}\}$ on level l. Hence, *algebraic smoothness* is defined as a generalization of the concept of geometric smoothness with the aim to extract some measurable quantity which can be (easily) computed from the matrix. In particular, in our application we weight the geometric smoothness of a surface (cf. Section 4) with the help of local surface classifier (cf. Section 3). Several different measures for algebraic smoothness are used today in the various algebraic multigrid methods [27,7, 9]. Common to all these heuristic definitions is the general observation that a simple relaxation scheme—most often Gauss–Seidel smoothing is used in AMG—damps (efficiently) high energy components, i.e. eigenvectors associated with large eigenvalues, only. Consequently, the coarse grid correction must be able to deal with the remaining small energy components. These small energy functions should be represented accurately on coarser grids.

The construction of the coarse basis $\{\Psi^{l,i}\}$ itself is a two-step process. First, we select so-called coarse grid points, i.e. a subset of indices which give the sparsity pattern of the prolongation matrix P^l. Then in a second step we define an interpolation formula, i.e. the weights of the prolongation matrix P^l. This tell us how a coefficient vector on a coarser level l is represented with respect to the finer level $l-1$. Thus, we define how information from the coarse basis $\{\Psi^{l,i}\}$ is represented in terms of the fine level basis $\{\Psi^{l-1,i}\}$. There are many different approaches to the definition of AMG prolongation matrixes. Our numerical experiments with different prolongation matrices showed that a renormalized variant of a very classical and widely used interpolation scheme, see e.g. [16], gave the most favorable results. Hence, throughout the paper we used this interpolation scheme with a simple averaging of the interpolation weights to enforce mass conservation. Note that this two-step process can also be viewed as a graph coarsening scheme: We select a subset of fine level vertices as the coarse vertex set and define an approriate sum of the weights of the removed edges on the fine level as weights for the coarse level edges (cf. Section 5).

To illustrate the performance of AMG, we give a here a very basic first example. Consider a flat square domain $\Omega = [-1;1]^2 \subset \mathbb{R}^2$. Now we select the subset $\Sigma = B_{1/2} - B_{1/2-\delta}$, where δ is a small positive real and B_r is the ball of radius r centered at the origin 0. Next, we define the function

$$\mathcal{C}(x) = \begin{cases} 2000, & x \in \Omega - \Sigma \\ 0.002, & x \in \Sigma \end{cases}$$

In other words, the operator C is smooth overall but exhibits a discontinuous jump on the ring-shaped boundary of Σ. Next, we define the following differential operator $\Delta_{\mathcal{C}} = -\mathrm{div}\,(\mathcal{C}\nabla\,\cdot)$. We discretize this problem by the usual finite element procedure. Hence, we define a quadratic form $a(\varphi,\psi) = \int_\Omega \mathcal{C}\ \nabla\varphi, \nabla\psi)$ corresponding to this operator. Then, let $\mathcal{V}_h$ be the finite element space corresponding to a triangulation of Ω and $\{\Phi^1, \ldots \Phi^n\}$ the basis of hat shaped basis functions, where n is the number of nodes of the triangulation. Finally, we compute the $n \times n$ finite element stiffness matrix A:

$$A_{ij} := a(\Phi_i, \Phi_j) = \int_\Omega \mathcal{C}\ \nabla\Phi_i \cdot \nabla\Phi_j\,.$$

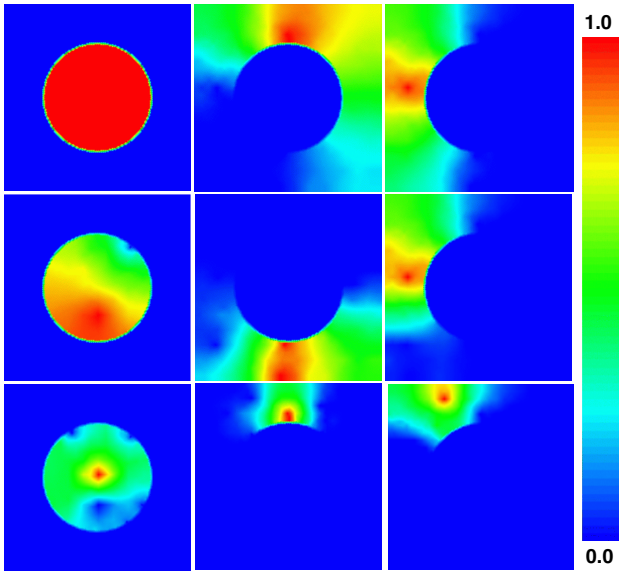

Fig. 5 For a simple second order differential operator on a planar domain, algebraic multigrid basis functions are depicted on different scales (upper row: coarsest scale, middle row and lower row successively finer scales). The basis functions clearly reflect the ring type feature region encoded in the operator.

The multiscale of AMG basis functions is depicted in Fig. 5. These basis functions clearly follow the discontinuities of C. However, note that in smooth regions the bases have a nonzero overlap. Moreover, the AMG method does not impose any constraints on the way this overlap takes place - for instance, it does not guarantee that a smooth region is entirely covered by a single basis function or by a number of bases having the same nonzero support size. Nevertheless, this is not a serious problem for our method (see, for more details, Sec. 5).

Obviously, the above is just a succint presentation of the AMG method. However, we stress again that AMG tools have been developed with the very purpose of being used as *black box* solvers. Since our method does not explicitly rely on the specific parameters or coarsening strategy of a given AMG solver, one should be able to easily substitute the AMG solver one avails of instead of the one we used, and obtain similar results. Different AMG parameter settings and coarsening strategies are likely to deliver slightly different basis functions, especially in the smooth areas. However, given the strong classifier discontinuities following the edge features, various AMG tools should deliver the same basis function behavior along these features.

3 Moment-based surface analysis

In the following, we will introduce and discuss local surface classification based on zero and first order surface moments.

This will in particular allow to robustly distinguish smooth regions from the vicinity of edges or corners on surfaces. For a surface $\mathcal{M}$, the zero moment is given by the local barycenter of $\mathcal{M}$ with respect to an Euclidian ball $B_\epsilon(x)$ centered at x:

$$M^0_\epsilon(x(\xi)) := \fint_{B_\epsilon \cap \mathcal{M}} x \, \mathrm{d}x \, .$$

The parameter ϵ serves as a filter width. Furthermore, the first moment is defined by

$$\begin{aligned} M^1_\epsilon(x) &:= \fint_{B_\epsilon(x) \cap \mathcal{M}} (x - M^0_\epsilon(x)) \otimes (x - M^0_\epsilon(x))) \, \mathrm{d}x \\ &= \fint_{B_\epsilon(x) \cap \mathcal{M}} x \otimes x \, \mathrm{d}x - M^0_\epsilon(x) \otimes M^0_\epsilon(x) \end{aligned} ,$$

where $y \otimes z := (y_i z_j)_{i,j=1,\dots,3}$. Due to the definition via local integration, the zero and the first moment is expected to be robust with respect to noise.

Moments in smooth areas and at edges

In the following two sections we will explain, how zero and first moment information may be used to distinguish between smooth and non-smooth surface parts. It turns out, that the zero moment shift, defined by

$$N_\epsilon(x) = M^0_\epsilon(x) - x \, ,$$

scales quadratically w.r.t. the filter width ϵ in smooth surface domains, whereas on edges and corners, the scaling is only linear (cf. Fig. 6). Furthermore, the eigenvalues of the first moment $\mathcal{M}^1_\epsilon(x)$ give us additional information in the presence of an edge. This justifies the usage of moments as detectors for surface features. For a given, usually small, parameter ϵ, only features larger than ϵ will be detected. The zero moment shift N_ϵ plays the role of a scaled approximate normal.

Indeed, the quadratic scaling of the zero moment is given by the relation

$$N_\epsilon(x) = -\epsilon^2 c(d) H(x) N(x) + o(\epsilon^2) \, .$$

The explicit constant c(d) = c(2) = 0.125 (we consider 2D surfaces only). The quantity $H(x)N(x)$ is the mean curvature vector at x. For a proof we refer to [12].

We now discuss the case of non-smooth surface features, such as edges and corners. Let $\mathcal{M}$ be a surface, which is smooth up to the edge set $\Sigma_\mathcal{M}$ on the surface. Then, for $X \in \Sigma_\mathcal{M}$, there is a vector $\tilde{N}(x)$, such that

$$N_\epsilon(x) = \epsilon \, \tilde{N}(x) + o(\epsilon) \, .$$

In this sense, the zero order moment scales linearly on the singularity set of the surface. Next, we consider the first moment. Let us assume that for $X \in \Sigma_\mathcal{M}$ the apex angle of a surface edge is of size 2φ (cf. Fig. 6). Then in ϵ the eigenvalues of the first moment are $\epsilon^2\gamma$, $\epsilon^2\gamma \, \sin^2 \varphi$ and $\epsilon^2\delta \, \cos^2 \varphi$ up to higher order terms, where $\gamma = 0.25$ and $\delta \approx 0.0699$. For a formal proof, we refer again to [12].

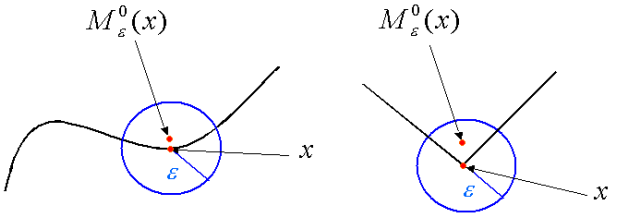

Fig. 6 The intersections of a ball $B_\epsilon(x)$ are drawn for points x in a smooth areas and on an edge respectively. In addition, we show the approximate normal $N_\epsilon(x)$ and the eigen direction of the first moment $M^1_\epsilon(x)$ for a point x on an edge.

Local surface feature classification

We will use these results to define local surface classifiers, i.e. quantities that enable us to robustly distinguish between smooth surface areas and features such as edges and corners. This will later be encoded in a mathematical operator on the surface (see Section 4). We have seen that the shift of the zero moment N_ϵ differs by an order of magnitude in ϵ if compared on edges and in smooth areas on the surface, respectively. Hence, let us define a first local surface classifier

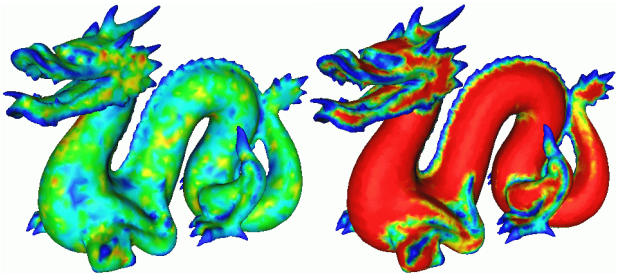

Fig. 7 The zero moment and the combined moment feature classifiers are compared on a triangular mesh. The combined classifier detects significantly more robustly the surface edges.

$$\mathcal{C}^0_\epsilon(x) = G\left(\frac{\|N_\epsilon(x)\|}{\epsilon}\right)$$

where $G(s) = \frac{1}{\alpha + \beta s^2}$ with suitably chosen $\alpha, \beta > 0$. In all our applications we have chosen $\alpha = 0.002$ and $\beta = 20$. We observe that $\mathcal{C}^0_\epsilon(x) \approx 1/\alpha$ in smooth regions on $\mathcal{M}$ and $\mathcal{C}^0_\epsilon(x) << 1/\alpha$ close to edges or corners (cf. Fig. 7). Even though $\mathcal{C}^0_\epsilon$ can already serve as a good classification tool, we can further improve the feature detection quality by incorporating first moment information. Suppose $\lambda_{min}, \lambda_{max}$ to be the smallest and largest eigenvalue of $M^1_\epsilon(x)$, respectively. From (1) we know that the quotient $\lambda_{min}/\lambda_{max}$ is approximately given by

$$\lambda_{min}/\lambda_{max} \approx \delta/\gamma \, \cos^2 \varphi \approx 0.2796 \cos^2 \varphi \, ,$$

where 2φ is the apex angle of an edge feature. This relation for $\lambda_{min}/\lambda_{max}$ is valid for φ larger than $0.2726 \approx 16^o$. Especially, in the smooth case ($\varphi = \pi/2$), this quotient vanishes where it increases for decreasing φ. Hence, we can further pronounce edges in the classification by the choice of a

combined zero and first moment classification

$$\mathcal{C}_\epsilon^{0,1} = G\left(\frac{\|N_\epsilon\|\,\lambda_{min}}{\epsilon\,\lambda_{max}}\right).$$

We mention that, for φ smaller than 16^o, the quotient of the eigenvalues again tends to 0, when $\varphi \to 0$. In this sense, very sharp features are detected in a weaker sense than they should. However, as our experiments showed, this seems to be only of theoretical interest. Figure 7 compares the results obtained by the classification with $\mathcal{C}_\epsilon^0$ and $\mathcal{C}_\epsilon^{0,1}$. For all surfaces we tried, the combined classifier showed a better separation of the feature areas (edges and corners) from the smooth areas than the zero moment classifier. Due to its superior quality we have applied the classifier $\mathcal{C}_\epsilon^{0,1}$ in all applications below (cf. Fig. 8).

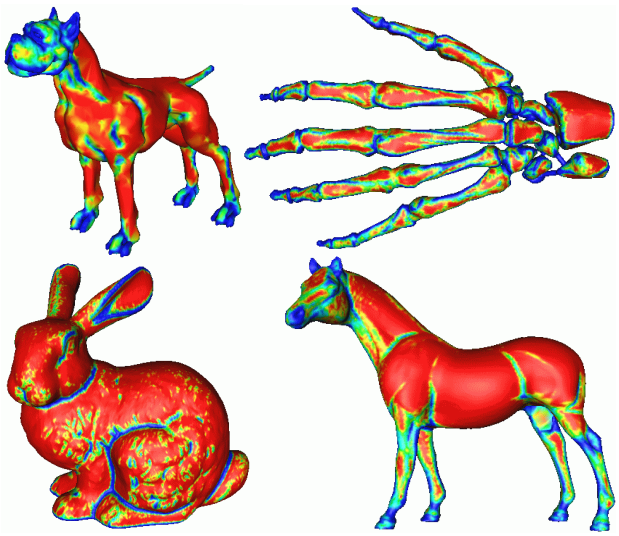

Fig. 8 For different triangular surface meshes we show the local feature classification result using a color coding for the classifier $\mathcal{C}^{0,1}(\cdot)$.

Implementation of zero and first moment

In the previous section, we have treated arbitrary surfaces. In applications, we usually deal with two-dimensional, irregular, triangular grids. In the following we will detail the discretization of the presented local surface classification in this case. We consider a triangular mesh $\mathcal{M}_h$ with grid size function h. In our implementation, we compute the moments centered at each node of the triangulation.

Let us fix one node X_i and denote the discrete moments by $M^0_{\epsilon,h}$ and $M^1_{\epsilon,h}$. Given this radius ϵ, we first collect all triangles $\{T_1, \dots T_m\}$ of the triangulation such that $T_i \cap B_\epsilon(X_i) \neq \emptyset,\ i = 1, \dots m$, by performing a simple breadth first search from the node X_i on the mesh connectivity graph. This set of triangles splits into two subsets. The first one - denoted by $\mathcal{T}^o$ - consists of all elements with $T_i \subset B_\epsilon(X_i)$. The second one $\mathcal{T}^\partial$ is the complement. Now we iteratively compute the integrals $⨍_{\mathcal{T}^o} x\,dA$ and $⨍_{\mathcal{T}^o} x \otimes x\,dA$. On each triangle of $\mathcal{T}^0$ we use the following exact integration formulas:

$$M^0(T_i) = \frac{1}{3}(X_0 + X_1 + X_2)\,,$$
$$⨍_{T_i} x \otimes x\,dA = \frac{1}{3}(Y_0 \otimes Y_0 + Y_1 \otimes Y_1 + Y_2 \otimes Y_2)\,,$$

where X_0, X_1, X_2 are the nodes of T_i and $Y_0 = (X_0 + X_1)/2$, $Y_1 = (X_1 + X_2)/2$ and $Y_2 = (X_0 + X_2)/2$. For the corresponding computations on $\mathcal{T}^\partial \cap B_\epsilon$ we apply an approximation. For each triangle $T_l \subset \mathcal{T}^\partial$, the intersection of the sphere ∂B_ϵ and the edges of the triangle consists of two points denoted by $P_1,\ P_2$. We replace the curvilinear connection $T_l \cap \partial B_\epsilon$ by the line segment connecting P_1 and P_2. Hence, we replace $T_l \cap B_\epsilon$ by a polygon which we again can split into triangles. We proceed now as above using exact integration on all these virtual triangles. To ensure a robust moment calculation we choose $\epsilon = 3\,h$ in our applications.

4 A matrix encoding features

Given a classifier $\mathcal{C} : \mathcal{M} \to \mathbb{R}_0^+$ on a surface $\mathcal{M}$, we can define a mathematical operator $\mathcal{A}[\mathcal{C}]$ which considers the classifier as a spatial coupling weight on the surface. Suppose $\mathcal{C}$ to be large in smooth surface regions and small on edges and corners. In our applications, we choose $\mathcal{C} = \mathcal{C}_\epsilon^{0,1}$ as above and define

$$\mathcal{A}[\mathcal{C}] := -\mathrm{div}_{\mathcal{M}}(\mathcal{C}\,\nabla_{\mathcal{M}}\,).$$

In case of a homogeneous surface with $\mathcal{C} = 1$ we obtain a constant spatial coupling described by the negative Laplace Beltrami operator $\Delta_M := -\mathrm{div}_{\mathcal{M}}\nabla_{\mathcal{M}}$. If one thinks in term of diffusion, $\mathcal{C}$ is the diffusion coefficient, which is small on edges and approximately $1/\alpha$ in smooth regions. This type of operator has already proved to be a powerful tool in feature preserving surface fairing and image denoising [26,11]. Here, we do not aim to process the surface via a differential equation. Instead we are interested in a multiscale decomposition of the operator itself. With respect to our actual aim of designing an editing tool for discrete, triangular surfaces instead of the continuous operator $\mathcal{A}$, we treat its discrete finite element counterpart $\mathcal{A}_h[\mathcal{C}]$. Hence, following the general finite element paradigms we first introduce the quadratic form $\mathcal{A}(\phi, \psi)$ acting on functions on $\mathcal{M}$:

$$\mathcal{A}(\phi, \psi) := \int_{\mathcal{M}} \mathcal{C}\ \nabla_{\mathcal{M}}\phi \cdot \nabla_{\mathcal{M}}\psi\ \mathrm{d}x\,.$$

Furthermore let

$$\mathcal{V}_h = \{\varphi_h \in C^0(\mathcal{M}_h)\ \big|\ \varphi_h|_T \in \mathcal{P}_1, T \in \mathcal{M}_h\}$$

be the finite element space on $\mathcal{M}_h$ consisting of those continuous functions being affine linear on each triangle of $\mathcal{M}_h$. The usual basis $\{\Phi_i\}_{i=1,\dots n}$, on $\mathcal{V}_h$ is defined by $\Phi_i(X_j) = \delta_{ij}$

where n is the number of vertices of $\mathcal{M}_h$ and $\Phi_i(X_j) = \delta_{ij}$ for all vertices X_j. Note that we use capital letters for discrete objects to distinguish them from continuous objects denoted with lower case letters. We now define a discrete operator $\mathcal{A}_h$ acting on $\mathcal{V}_h$ and a corresponding $n \times n$ matrix A where a matrix entry is given by

$$A_{ij} := \mathcal{A}(\Phi^i, \Phi^j) = \int_{\mathcal{M}_h} \mathcal{C}\ \nabla_{\mathcal{M}} \Phi_i \cdot \nabla_{\mathcal{M}} \Phi_j \ \mathrm{d}x$$

and $\{\Phi_1, \ldots, \Phi_J\}$ is the standard basis of $\mathcal{V}_h$. This matrix describes the coupling on the discrete surface weighted by the classifier $\mathcal{C}$. This coupling is encoded in terms of the coupling of adjacent nodes of the triangulation. Indeed, for every pair of adjacent nodes X_i and X_j the matrix entry A_{ij} describes the coupling strength. In Section 5 we will discuss the multiscale decomposition of this matrix, the centerpoint of our method.

Assembling the matrix

Before we discuss the multiscale decomposition of the matrix A, we detail its actual computation. The assembly of A is based on the standard Finite Element assembly procedure. We start by initializing $B = 0$ followed by a traversal of all surface triangles T. On each T with nodes P^0, P^1, P^2, a corresponding local matrix $(a_{ij}(T))_{ij}$ is computed first, corresponding to all pairings of local nodal basis functions. Next, the local matrix is added to the matching locations in the global matrix B, i. e. for every pair i, j we update $A_{\alpha(i),\alpha(j)} = A_{\alpha(i),\alpha(j)} + a_{ij}(T)$. Here $\alpha(i)$ is defined as the global index of the node with local index i. For the local matrix we need a local surface classifier $\mathcal{C}(T)$ for every triangle T on $\mathcal{M}_h$, which we define by averaging. We obtain for the local matrix:

$$\begin{aligned} a_{ij}(T) &= \mathcal{C}(T) \int_T \nabla_T \Phi_i \cdot \nabla_T \Phi_j = \mathcal{C}(T)\,|T| \frac{\nu_i}{h_i} \cdot \frac{\nu_j}{h_j} \\ &= \mathcal{C}(T)|T| \frac{e_i}{h_i\ \|e_i\|} \cdot \frac{e_j}{h_j\ \|e_j\|} = \mathcal{C}(T) \frac{e_i \cdot e_j}{4|T|} \end{aligned}$$

where $|T|$ is the area of triangle T, Φ_i is the nodal basis function corresponding to the node x_i for any local index i, ∇_T the gradient on T, and ν_i the outer normal to the edge e_i opposite of x_i. Finally h_i is the height of the triangle over the edge e_i. The trianglewise classifier $\mathcal{C}(T)$ is deduced by averaging from the classifier values on the nodes P^1, P^2 and P^3 of the triangle T: $\mathcal{C}(T) = 1/3(\mathcal{C}(P^0) + \mathcal{C}(P^1) + \mathcal{C}(P^3))$. Given the sparsity of A, we use a compressed row matrix storage model, i.e. store only the nonzero entries and their column indexes, for every matrix row. This confines the matrix memory requirements to e.g. around 10 megabytes for a mesh of 280472 triangles.

5 Multiscale decomposition by AMG

As discussed so far, the matrix A defined above can be regarded as a description of the surface shape. In particular the smoothness modulus and the distinct surface features are encoded in this matrix. Besides prominent feature edges, successively finer, more detailed edges are encoded. At this point, we require a tool capable to analyze and represent this multiscale of features. Here AMG comes into play. Given a matrix which encodes inhomogeneities on different scales - in our case the features detected by the classifier - we apply AMG (cf. Section 2) to detect this multiscale. AMG will deliver a scale of surface descriptions in terms of matrices A^l for $l = 0, \cdots L$ ranging from detailed ($A^0 = A$) to very coarse (A^L). Together with these matrices we obtain basis functions $\Psi^{l,i}$ on all scales. Hence, we obtain handles for surface editing on different detail scales. One might either manipulate large scale features such as the head, tail, or legs of the meshes shown in this paper. Alternatively, adjustments of small details, such as finger tips or ears, can be performed. This section describes the underlying mathematics related to the multiscale representation. The next section presents the actual editing tool, configured as a simple but effective "combiner" of basis functions.

Recalling, we apply the AMG algorithm (cf. Section 2) to the matrix $A \in \mathbb{R}^{n,n}$ introduced in the previous section. Running AMG on the matrix A we obtain a sequence of prolongation matrices

$$P^l \in \mathbb{R}^{n_{l-1}, n_l}\ .$$

as output, where $\{n_l\}_{l=0,\cdots,L}$ is decreasing and $n_0 = n$. The entries in each column of P^l describe how the basis functions $\Psi^{l,i}$ for $i = 1, \cdots, n_l$ can be generated from the basis functions $\Psi^{l-1,i}$ for $i = 1, \cdots, n_{l-1}$ on the previous, finer level. Indeed, we obtain the following simple recursive recipe to calculate a multiscale of basis functions

$$\begin{aligned} \Psi^{l,i} &:= \sum_{j=1,\cdots,n_{l-1}} P^l_{ji} \Psi^{l-1,j} \quad \forall i = 1, \cdots n;\ l = 1, \cdots L \\ \Psi^{0,i} &:= \Phi^i \quad \forall i_1, \cdots n \end{aligned}$$

Collecting all basis function $\Psi^{l,i}$ on all scales $l = 0, \cdots L$ we build up a *multiscale library*

$$\mathcal{L}(\mathcal{C}) = \{\Psi^{l,i}\}_{\substack{l=0,\cdots,L \\ i=1,\cdots,n_l}}$$

of functions which reflects, on all scales, the surface features encoded by the local surface classifier $\mathcal{C}$ (cf. Fig. 9). Let us recall that the prolongation matrices induce a sequence of matrix representations $A^l \in \mathbb{R}^{n_l, n_l}$ on different levels:

$$\begin{aligned} A^l &:= R^l A^{l-1} P^l \quad l = 1, \cdots, L \\ A^0 &:= A \end{aligned}$$

where the restriction matrices $R^l \in \mathbb{R}^{n_{l-1}, n_l}$ are defined as $R^l = (P^l)^T$. In general, as outlined in Sec. 2 the goal of AMG is to compute prolongations in such a way that, for the number of degrees of freedom n_l, the mapping corresponding to the matrix A^l is a sufficiently good approximation of the original matrix A. Hereby, the underlying algebraic smoothness criterion depends on the problem setting. In our case, smoothness is induced by the spatially varying surface classifier $\mathcal{C}(\cdot)$. An interpretation of the entries of A^l is that A^l_{ij} measures of

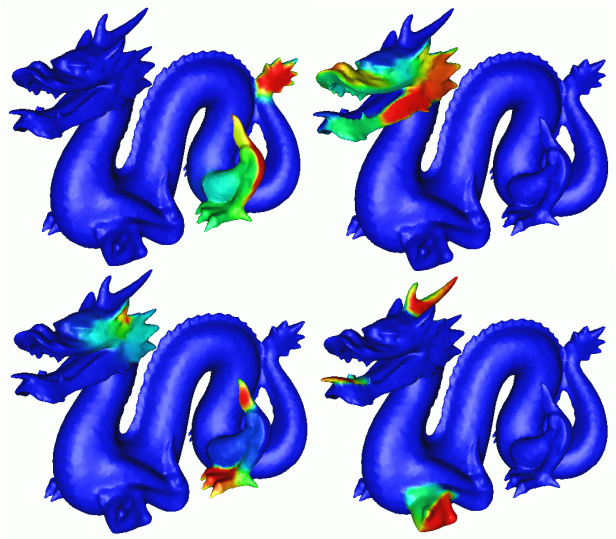

Fig. 9 Selected basis functions $\Psi^{l,i}$ are color coded on a blue (low) to red (high) colormap on the coarsest scale (upper row) and on the third coarsest scale(lower row).

strength of the coupling between the basis functions $\Psi^{l,i}$ and $\Psi^{l,j}$ or - if we think in term or surface regions - the coupling of the domains defined by the supports of the basis functions. In particular, the coupling is expected to be weak across edges, as described by $\mathcal{C}(\cdot)$.

Furthermore, the shape of the basis functions will clearly show the strength of the node coupling in the matrix. On edges, the weights A_{ij} are small, because the classifier $\mathcal{C}(\cot)$ is small in this region. Hence, AMG will cluster vertices on both sides of an edge on much finer scales and will collect vertices from both sides of the edge at later stages of the coarsening process. In particular, it is expensive - in terms of the built-in optimization in a concrete AMG implementation - to generate basis functions whose masses are equally distributed on different sides of an edge feature (cf. Fig. 5 and 9). However, as already mentioned in Sec. 2, this is not a problem for the proposed method, as it will be explained next.

As usually, basis functions on a given scale overlap each other. Hence, it turns out to be sometimes hard to treat to visualize basis functions directly in an graphical user interface for e.g. a surface editor or processing tool. Hence, aiming to represent the set of overlapping basis functions $\{\Psi^{l,i}\}_{i=1,\cdots,n_l}$ visually, let us define a corresponding domain decomposition $\mathcal{D}^l$ for every $l = 0, \cdots L$ (cf. Fig.10). Here, we define $\mathcal{D}^l := \{\mathcal{D}^{l,i}\}_{i=1,\cdots,n_l}$, where

$$\mathcal{D}^{l,i} := \{x \in \mathcal{M}_h \,|\, \Psi^{l,i} \geq \Psi^{l,j} \,\forall j = 1, \cdots, n_l\}$$

Let us remark that the domains on different scales need not be strictly spatially nested. Nethertheless, these domains are bounded by surface feature lines. This characteristic is enough for building a simple and intuitive way to represent and manipulate such domains on different scales (see Sec. 6).

A major feature of our method is its speed. The AMG computation of the prolongation matrices takes between 3 and 6 seconds for meshes up to 300000 elements on a Pentium 4 PC at 1.5 GHz running Linux. The domain decomposition involves just multiplication of the prolongations and thus takes, for the same datasets and platform, 1 to 3 seconds. The slowest part of the pipeline is assembling the classifier matrix, which is linear in the number of mesh triangles, and takes about 10 seconds for the largest mesh we tried, i.e. 280472 elements. The matrix assembly complexity is quadratic in the radius ϵ of the integration ball B_ϵ (see Sec. 3). For all examples, a ball size of $3h$, where h is the average triangle size, was used. Larger balls, that would slow down the assembly, are not required, as the surface features we are looking for in the classifier are already present on this scale.

The graph perspective

The mesh $\mathcal{M}^0 := \mathcal{M}_h$ can be trivially encoded in a graph $\mathcal{G}^0 = \mathcal{G}(\mathcal{N}^0, \mathcal{E}^0)$, where the vertices $\mathcal{N}^0 := \mathcal{N}_h$ and the egdes $\mathcal{E}^0 := \mathcal{E}_h$ of the mesh are the graph nodes and egdes, respectively. In case of our affine finite element space $\mathcal{V}_h$ the sparsity pattern of the matrix A^0 is such that in the ith row, corresponding to the vertex x_i, the only nonzero entries A^0_{ij} are those corresponding to adjacent nodes x_j, connected to x_i by an edge $e^0_{ij} \in \mathcal{E}^0$, and the entry x_i reflecting the self-coupling of node i with itself. Hence, the entries A_{ij} in the matrix can be regarded as weights on the egdes $\mathcal{E}^0$ of the graph $\mathcal{G}^0$. Indeed, AMG generates a sequence of graphs

$$\mathcal{G}^l = \mathcal{G}(\mathcal{N}^l, \mathcal{E}^l)$$

for $l = 1, \cdots, L$. On level l the set of graph nodes $\mathcal{N}^l$ corresponds to the basis $\{\Psi^{l,i}\}_{i=1,\cdots,n_l}$ and for every entry $A^l_{ij} \neq 0$ their exists an edge $e^l_{ij} \in \mathcal{E}^l$ with that weight. One might ask whether the graphs $\mathcal{G}^l$ for $l > 0$ again generate immersed polygonal grids $\mathcal{M}^l$. This is known to be a design principle of progressive mesh algorithms. However, in our case there is in general no such mesh nesting sequence and it would be a much too severe restriction to formulate this property in the AMG algorithm as a constraint.

6 A multiscale surface editor

The basis functions $\Psi^{l,i}$ from the multiscale library $\mathbf{L}(\mathcal{C})$ can be directly used as handles to process the surface. Frequently, however, the "handles" the user has in mind to manipulate the surface are not precisely recovered by one of the available AMG basis functions. Desired handles can be generated by combining a few basis functions from the AMG multiscale library of basis functions. We present here a simple but effective feature editor based on this strategy. The editor allows to select a given basis function on a given level by just two intuitive mouse picks. Several such bases can be then added to design the desired handle. In detail, for an arbitrary surface point x - chosen by a first mouse pick - we extract from the multiscale domain decomposition a sequence of activated domains $\{\mathcal{D}^l_x\}_{l=0,\cdots,L}$, where $\mathcal{D}^{l,i(x)}$ is the set $\mathcal{D}^{l,i}$ from the domain decomposition on level l for which $x \in \mathcal{D}^{l,i}$. A second point

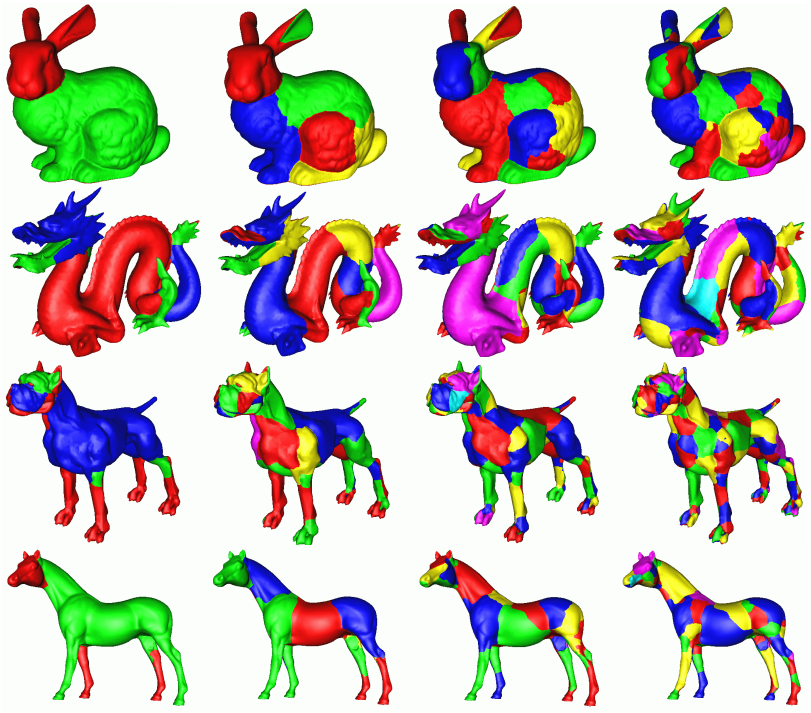

Fig. 10 On different scales (corresponding to the columns) the domain decomposition $\mathcal{D}^l$ is shown for several triangular surfaces. The surfaces consist of 280472 (bunny), 87140 (dragon) 25030 (hound), and 96966 (horse) triangles, respectively.

y - chosen by a second mouse pick - identifies now a single active set $\mathcal{D}^{l(y),i(x)}$ from the activated sets, where $l(y) = \max\{l \,|\, y \in \mathcal{D}^{l,i(x)}\}$. Hence, the corresponding basis function $\Psi^{l(y),i(x)}$ is interactively and intuitively selected form the multiscale library $\mathcal{L}(\mathcal{C})$. This function inintializes the handle

$$\Psi \leftarrow \Psi^{l(y),i(x)} .$$

The process can be repeated e.g. by picking another two surface points $\tilde{x}$ and $\tilde{y}$. The handle is updated

$$\Psi \leftarrow \min\{1, \Psi + \Psi^{l(\tilde{y}),i(\tilde{x})}\} .$$

Figure 11 shows an example. The first pick (at the black arrow's location) produces the activated domains corresponding to the hand's middle finger tip. Surface color coding indicates the editor's current status. Picking a point x, the activated domains are drawn in colors corresponding to the sequence parameter l, using a fixed color map (see Fig. 11). The second pick (light blue arrow) adds now a basis to the current handle. The domain

$$\mathcal{D}_\delta = \{z \in \mathcal{M}_h \,|\, \Psi(z) > \delta\}$$

where we choose $\delta = 0.01$, essentially being the support of the current handle Ψ, is always shown by a fixed color (light blue, Fig. 11). Coloring guides the user's iterative handle selection. In all our applications, 1 to 5 selection iterations (i.e. 2 to 10 clicks) were sufficient to define the desired surface handles (cf. Figures 11,12, 13, 14). In addition, we provide a mechanism to step back in the handle construction. Picking a point x in the already selected domain $\mathcal{D}$ deletes the previously added basis function containing x in its support.

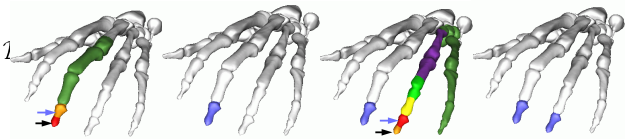

Fig. 11 Selection steps in the feature editor (from left to right). First, for a picked point x, all active domains $\mathcal{D}^{l,i(x)}$ are color coded orange, red, yellow, green and violet for increasing scale l. Next, picking into one of these active domains at point y, selects a particular scale $l(y)$. This activates $\mathcal{D}^{l(y),i(x)}$ and the corresponding basis function $\Psi^{l(y),i(x)}$. The support is drawn in blue. Repeating this procedure adds a second basis function to the handle with two clicks.

7 Applications

After having selected the desired surface handles, one can edit the surface. We show here a number of simple surface editing operations performed on the selected handles. These operations serve only as illustration for the presented multiscale surface decomposition and handle construction. However, this does not diminish the usability of our technique. Indeed, state of the art surface processing operations can be easily substituted in place of the ones shown here.

In the first example (Fig. 12), we select five features on the Stanford bunny dataset, i.e. the ears, from paws, and tail. Using the handle construction method (Sec. 6) these features are easily selected by just ten mouse clicks, two for every feature, in the order: left ear, right ear, left paw, right paw, tail (Fig. 12 a-j). The complete handle is shown in Fig. 12 k. Next, we remove the ears by smoothing the mesh. Smoothing deforms the mesh in the inward normal direction and performs

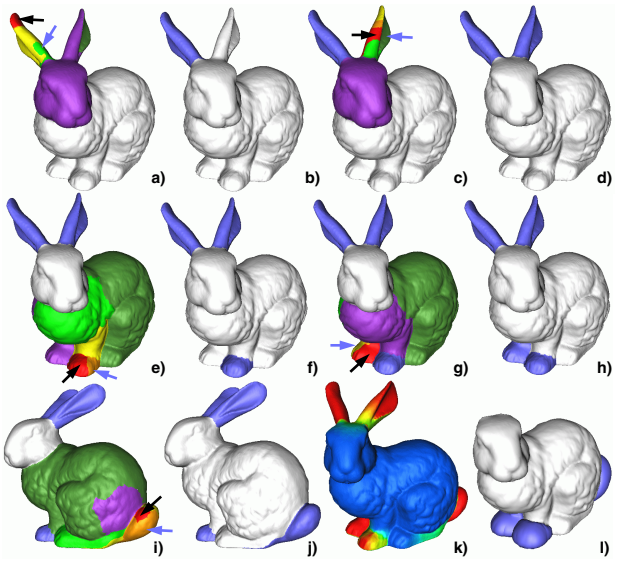

Fig. 12 Application 1: In ten clicks, five features (ears, front paws, tail) are selected (a-j). Next, the handle and the edited surface are shown (k,l)

a mesh decimation simultaneously by removing triangles that become smaller than a fraction of the average triangle size. Decimation is needed to ensure that the deformed mesh does not contain unnecessarily small triangles. Finally, we inflate the paws and tail by mesh deformation in the outward normal direction. Figure 12 l shows the edited mesh and the selected domain.

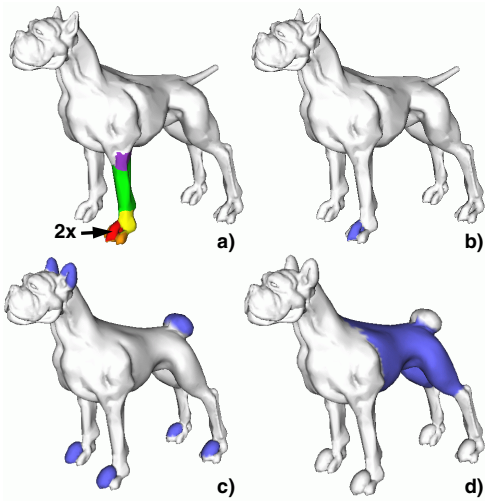

Fig. 13 Application 2: The left hind leg middle toe is selected by two clicks on the red domain (a,b). Similarly, other features are selected (c). Finally, the selection is edited (d)

In the second example, we select eight features on the mesh in Fig. 13, i.e. the ears, middle toes, and tail. Selecting these fine details requires only two clicks per feature. Figures 13 a,b show the selection of the left front toe. Next, we inflate the toes and ears and round the tail. (Fig. 13 c). The inflation is done as for the previous example. The tail rounding is a sequence of alternate mesh inflations and smoothings. Finally, we separately select the body, also in two clicks, and smooth it. Figure 13 d shows the final result and the domain corresponding to the body.

In the last example, we select four features on the dragon dataset (Fig. 14), i.e. the horn, tongue, hind leg spike, and tail tip. We detail the selection of the horn. The first click (Fig. 14 a) produces the activation domains for the horn's tip. A second click, in the same place, selects the upper half of the horn only, since there is no single basis covering the whole horn (Fig. 14 b). Two more clicks, both on the horn's stem, are needed to select the rest of the horn (Fig. 14 c,d). After all details are selected (Fig. 14 e), we erase them by mesh decimation, to yield the final result (Fig. 14).

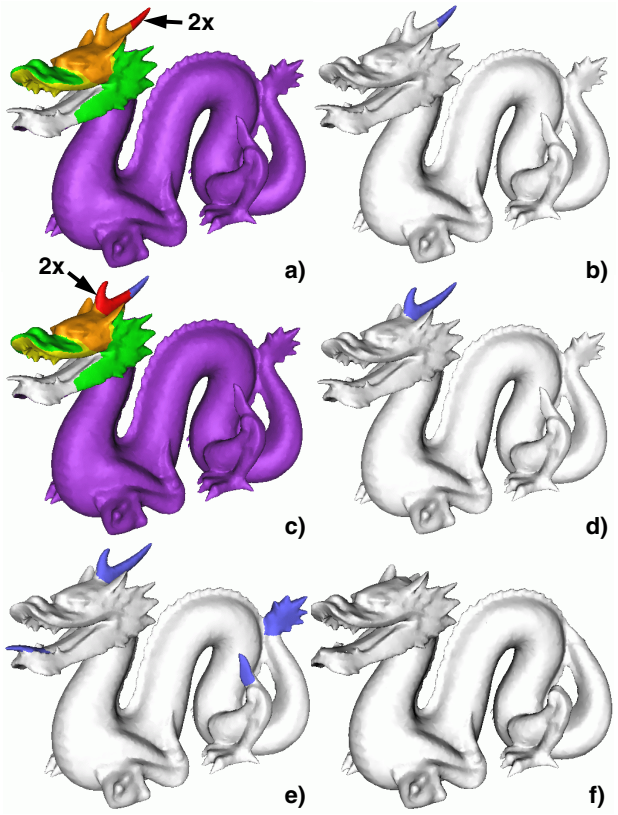

Fig. 14 Application 3: The dragon's horn is selected in a sequence of four clicks (a..d). Next, other features are selected (e). Finally, the selection is erased (f)

8 Conclusions

We have presented a novel technique for manipulating surface meshes at different levels of detail, consisting of the following ingredients: the stable computation of surface classifiers, the classifier assembly into a finite element matrix, the computation of a sequence of basis functions on different detail levels with the AMG method, and a simple but effective surface editor based on these basis functions. Overall, selecting surface features at different detail levels is done by a few mouse

clicks. Although the machinery behind the editor is quite involved, its users may employ it being totally unaware of the underlying complexities.

The whole process requires setting few (if any) parameters. The two classifier parameters α and β (Sec. 3) were fixed for all our test surfaces. The AMG tool specific parameters were fixed as well for all surfaces. There is little else that could be automated in the process. The most complex implementation part of the entire pipeline is indeed the AMG tool. However, as mentioned, several available AMG tools can be used, virtually as black boxes. Implementing the moment-based classifiers, matrix assembly, basis function computation, and the editor, is straightforward.

The presented technique opens a multitude of directions for surface processing. Various other data, such as surface parameterizations, texture, shading, or normals can be represented on the multiscale induced by surface features. State of the art surface processing operations, such as editing, decimation, or morphing, can be coupled with the surface decomposition output. Such surface data can also be encoded into new classifiers, to produce novel ways for multilevel surface representations. Finally, an interesting question is whether the presented AMG-based technique can be applied to mesh-free, point-based surface representations, such as the one used in the PointShop 3D editing tool [36].

References

1. L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Arch. Ration. Mech. Anal.*, 123 (3):199–257, 1993.
2. H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. *Computer Graphics Proceedings (SIGGRAPH '01)*, pages 185–194, 2001.
3. A. Brandt. Algebraic Multigrid Theory: The Symmetric Case. In *Preliminary Proceedings for the International Multigrid Conference*, Copper Mountain, Colorado, April 1983.
4. A. Brandt. Algebraic Multigrid Theory: The Symmetric Case. *Appl. Math. Comput.*, 19:23–56, 1986.
5. A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic Multigrid for Automatic Multigrid Solutions with Application to Geodetic Computations. Technical Report, Institute for Computational Studies, Fort Collins, Colorado, October 1982.
6. A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic Multigrid for Sparse Matrix Equations. In D.J. Evans, editor, *Sparsity and Its Applications*. Cambridge Univ. Press, 1984.
7. M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Algebraic Multigrid Based on Element Interpolation (AMGe). *SIAM J. Sci. Comp.*, 22(5):1570–1592, 2000.
8. V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numer. Math.*, 66, 1993.
9. T.P. Chartier. Spectral AMGe (ρAMGe). *Abstracts of the Seventh Copper Mountain Conference on Iterative Methods*, 2, 2002.
10. B. Chazelle, D.P. Dobkin, N. Shouraboura, and A. Tal. Strategies for Polyhedral Surface Decomposition: An Experimental Study. In *Computational Geometry, Theory and Applications*, volume 7 (4-5), pages 327–342, 1997.
11. U. Clarenz, U. Diewald, and M. Rumpf. Nonlinear anisotropic diffusion in surface processing. In *Proc. IEEE Visualization*, pages 397–405. IEEE CS Press, 2000.
12. U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. *to appear in Transactions on Visualization and Computer Graphics*, 2004.
13. R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *Int. Jr. Comp. Vis.*, 1:167–187, 1987.
14. M. Desbrun, M. Meyer, P. Schroeder, and A. Barr. Anisotropic feature preserving denoising of height fields and bivariate data. In *Graphics Interface '00 Proceedings*. AK Peters Ltd., 2000.
15. M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, Boston–Basel–Berlin, 1993.
16. T. Grauschopf, M. Griebel, and H. Regler. Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic Multigrid Coarsening for Second Order Elliptic PDEs. *Applied Numerical Mathematics*, 23(1):63–96, 1997.
17. A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based Surface Decomposition for Correspondence and Morphing between Polyhedra. In *Proc. of Computer Animation*, pages 64–71, 1998.
18. I. Guskov, W. Sweldens, and P. Schroeder. Multiresolution Signal Processing for Meshes. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, 1999.
19. A. Hubeli and M.H. Gross. Multiresolution Feature Extraction from Unstructured Meshes. In *Proc. IEEE Visualization*, pages 287–294. IEEE CS Press, 2001.
20. L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 105–114, 1998.
21. Mundy J. L. and Zisserman A. *Geometric Invariance in Computer Vision*. MIT Press, 1992.
22. A. Lee, W. Sweldens, P. Schroeder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parametrization of surfaces. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 95–104, 1998.
23. B. Lévy, S. Petitjean, Ray N., and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Computer Graphics (SIGGRAPH '02 Proceedings)*, pages 362–371, 2002.
24. A.P. Mangan and R.T. Whitaker. Partitioning 3D Surface Meshes Using Watershed Segmentation. In *IEEE TVCG*, volume 5 (4), pages 308–321, 1999.
25. H.P. Moreton and C.H. Séquin. Functional optimization for fair surface design. In *Proc. ACM SIGGRAPH*, pages 167–176, 1992.
26. P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In *IEEE Trans. Patt. Mach. Intell.*, volume 12(7), pages 629–639, 1990.
27. J. W. Ruge and K. Stüben. Efficient Solution of Finite Difference and Finite Element Equations by Algebraic Multigrid. In D. J. Paddon and H. Holstein, editors, *Multigrid Methods for Integral and Differntial Equations*, The Institute of Mathematics and its Applications Conference Series. Clarendon Press, 1985.
28. E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *Proc. IEEE CVPR*, pages 70–77, South Carolina, 2000.
29. G. Taubin. *Recognition and Positioning of Rigid Objects using Algebraic and Moment Invariants*. PhD thesis, Brown University, Providence, RI, 1992.

30. U. Trottenberg, C. W. Osterlee, and A. Schüller. *Multigrid*, Appendix A: An Introduction to Algebraic Multigrid by K. Stüben, pages 413–532. Academic Press, San Diego, 2001.
31. J. Weickert. Foundations and applications of nonlinear anisotropic diffusion filtering. *Z. Angew. Math. Mech.*, 76:283–286, 1996.
32. J. Wu, S. Hu, C. Tai, and J. Sun. An effective feature-preserving mesh simplification scheme based on face constriction. In *Proc. Pacific Graphics 2001*, pages 12–21, 2001.
33. D. Harel Y. Koren, L. Carmel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In *Proc. IEEE InfoVis*, pages 137–144, 2002.
34. D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics Proceedings (SIGGRAPH 96)*, pages 259–269, 1997.
35. E. Zuckerberger, A. Tal, and S. Shlafman. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, 2002.
36. Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proc. Computer Graphics and Interactive Techniques*, pages 322–329. ACM Press, 2002.