Sketch-based Adaptive Mesh Augmentation using Stellar Operators

Afonso Paiva
ICMC–USP
apneto@icmc.usp.br

Ronan Amorin
University of Calgary
rmamorim@ucalgary.ca

Luiz Velho
IMPA
lvelho@impa.br

Mario Costa Sousa
University of Calgary
smcosta@ucalgary.ca

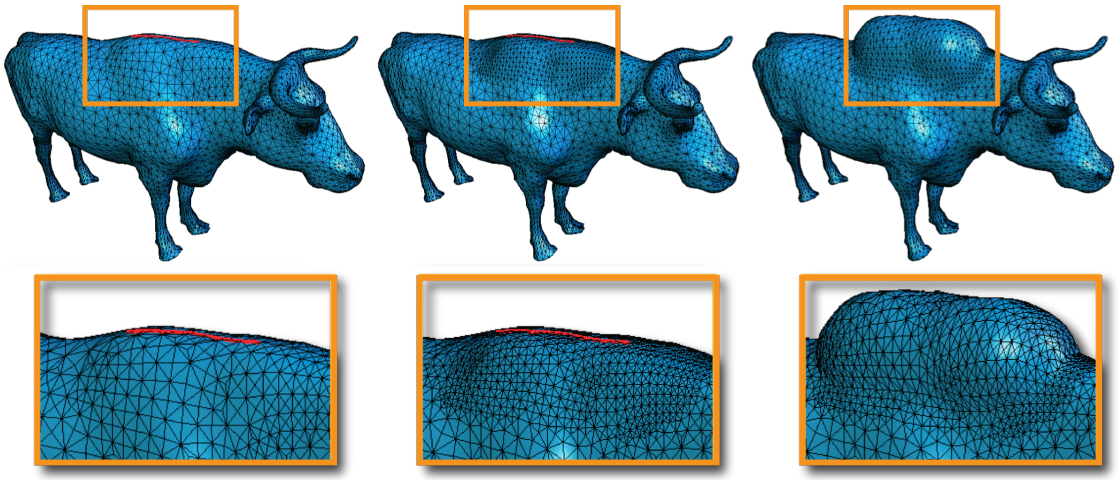


Fig. 1. Overview of our modeling framework: The sketching stage allows the graphic designer to place feature curves (red stroke) on the surface of the mesh (left); The adaptive mesh refinement stage increases the mesh resolution around the features (middle); Finally, the feature creation displaces vertices to create a surficial feature (right).

***Abstract*—In this paper we present a new method for modeling and editing surface detail using free-form curves and a natural interface. It combines in a original way an adaptive multiresolution mesh structure with a simple, intuitive sketch-based interface. One of the novel contributions of this work is the curve sensitive mesh resolution control, which allows the definition of a rich set of operators that locally modify the surface geometry. Furthermore, the present framework provides the basic functionality to build a complete feature based modeling system.**

***Keywords*- sketch-based modeling, adaptive meshes, sketching, silhouette, multiresolution, geometric modeling.**

I. Introduction

An important class of sketch-based interface and modeling (SBIM) operations is known as *surficial* augmentation. The goal is to allow users to add details over an existing model by sketching features on its surface. These features can represent creases, convex/concave regions, geometry representing a particular material property [1].

One important goal and challenge of surficial SBIM augmentation is to preserve the original modeling intent expressed by user's the 2D sketch input ("What You Sketch Is What You Get"). This must be efficiently and effectively mapped over the existing surface, resulting in new features that preserve the original quality of the mesh and the surficial augmenting intention.

Different surficial SBIM augmentation strategies have been proposed, including subdivision surfaces [2], [3], implicit surfaces [4] and silhouettes [5], [6].

A. Contributions

The main contribution of this paper is provide a simple and intuitive interface between sketch-based techniques and adaptive multiresolution meshes to create a robust and compact mesh augmentation system. In particular,

- We build on the adaptive multiresolution structure provided by semi-regular 4-8 meshes to create an unified framework for mesh augmentation using scheduled simplification, refinement stellar operations and different sketch styles to accomplish mesh augmentation;
- Our system allows to easily introduce new features on a mesh in an efficient manner. This efficiency is achieved through by coupling the sketching process into a mesh refinement process. During the mesh augmentation development, we adapt the mesh resolution only in the vicinity of the user's sketches; it allows the concentration of computational effort where it is needed;

- Our method introduces new strategies in the geometry adjustment stage, such as a vertex snapping process suited to semi-regular meshes and simple refinement criteria without compute intersections between edges and the user's sketches;
- We can localize the effects of the feature creation process, and adjusting vertices positions to create sharp features while elsewhere preserving the quality of the input mesh.

II. Related Work

Khodakovsky and Schröder [7] provide a method to enable the creation and control of fine-level feature lines using a non adaptive multiresolution mesh based on Loop's subdivision scheme. In order to accomplish the feature creation in this framework, it is essential to force the feature curves constrains at each level of the multiresolution tree. Both the surface and the feature curves are needed to represent the resulting surface. Finally, the entire mesh is subdivided one or more times to achieve quality mesh with the new features.

Biermann *et al.* [2] present a technique to create sharp features from feature strokes over quadrilateral subdivision surfaces. The input strokes are projected on the mesh's parameter space. Snapping is then processed in parameter space to align the mesh edges with the input strokes. A resampling is then performed so the augmentation can be seen in the actual mesh. However, in their framework, they create an adaptive quadrilateral mesh using a reparametrization process on Catmull-Clark surfaces.

Olsen *et al.* [3] present a method for augmenting an existing quadrilateral mesh with variable-scale sharp features. No special subdivision rules are required to enhance the augmented mesh. They use adaptive subdivision allowing the mesh complexity to increase only in the vicinity of the features, thus avoiding the need for global subdivision to create high-quality features. Their approach focus on Catmull-Clark surfaces with operators for interior feature lines augmentation. Our framework handles triangular meshes using stellar operators producing vertices with low valence. Our vertex snapping is performed in coarse resolution which automatically adapted to fine resolution without the need to recompute the entire process. Moreover, due to the stellar operators, the use of templates is not necessary.

Nealen *et al.* [5] present a view-dependent sketch-based approach for editing and deforming surface meshes. A set of vertices (handle) is selected by silhouette selection and cropping, or by sketching directly onto the surface. Editing is then processed by sketching a new, view-dependent handle position or by indirectly affecting differential properties along the sketched curve. The fundamental idea of their approach is to satisfy linear modeling constraints while preserving differential properties of the original mesh geometry. Their approach does not use any multi-resolution adaptive scheme. Shape features are generated by solving a linear system in the least-square sense. Our method is matrix-free and feature smoothing/sharpening is controlled by evaluating a simple scalar-valued function during vertex displacement. Our goal, however, is not to deform the mesh for preserving its local details but to augment it with new shape features and details.

Gingold and Zorin [8] present a sketch-based modeling technique based on editing shaded images of 3D models. The 3D models are automatically changed to reflect the modifications made in the 2D image. The features are included in the model by solving a quadratic surface optimization problem with linear constraints that approximates the user 2D stroke shading modification in the 3D model. Their approach uses an adaptive subdivision to refine the stroke area.

De Araújo and Jorge [9] present a system to create and edit free-form shapes from large point datasets. This is accomplished by re-polygonizing only the local changed parts of the model being created or edited. This approach relies on the curvature information extracted directly from the implicit surface. The Multi-level Partition of Unity Implicits (MPU) is used to convert point clouds into implicit surfaces. This MPU data structure was modified to allow local re-polygonizations and therefore permitting the handling of large datasets of points. Their approach is based on the re-polygonization of the implicit representation instead of working directly on the mesh as our approach.

Takayama *et al.* [10] present a 3D surface geometry cloning tool capable of cloning features of different topologies such as a handle, on high resolution meshes, in real-time using a brush. The tool allows the combination of features from different models filling gaps and cloning arbitrary surface geometric details. The proposed formulation guarantees a C^1 connection of the geometry cloned in the target surface. heir approach explores the regular 2D domain provided by local surface parametrization and solving in GPU the related variational problem.

III. The Method

Our mesh modeling framework consists of three stages (Figure 1). The sketching stage, described in Section III-A, handles the graphic designer's sketch input to create feature curves. The adaptive mesh refinement stage, detailed in Section III-B, uses the sketch input to increase the mesh resolution in the region of interest during the modeling of the features. Finally, the feature creation stage presented in Section III-C uses vertices in the high-resolution regions of the mesh to approximate each features, and then displaces these vertices to create surficial features on the mesh.

The input to our framework is a triangle mesh represented by a simplicial cell complex $M = (V, E, F)$ formed by a list of 3D vertices $\mathbf{v}_i = (v_i^x, v_i^y, v_i^z) \in V$, a list of edges $e = (\mathbf{v}_i, \mathbf{v}_j) \in E$ and a list of faces $f = (\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k) \in F$. We use a semi-regular 4-8 mesh structure based on stellar operations [11] for the adaptive mesh refinement stage. This allows us to increase the resolution of the mesh in a neighborhood around the features. After mesh refinement stage, features are created by displacing the surface along the sketch.

In our method, the surface displacement is made in two ways: the vertices in the neighborhood of the sketch are displaced along the normal direction or the vertices in the

neighborhood of the surface silhouette are displaced to new sketch shape for this silhouette (see Figure 2).

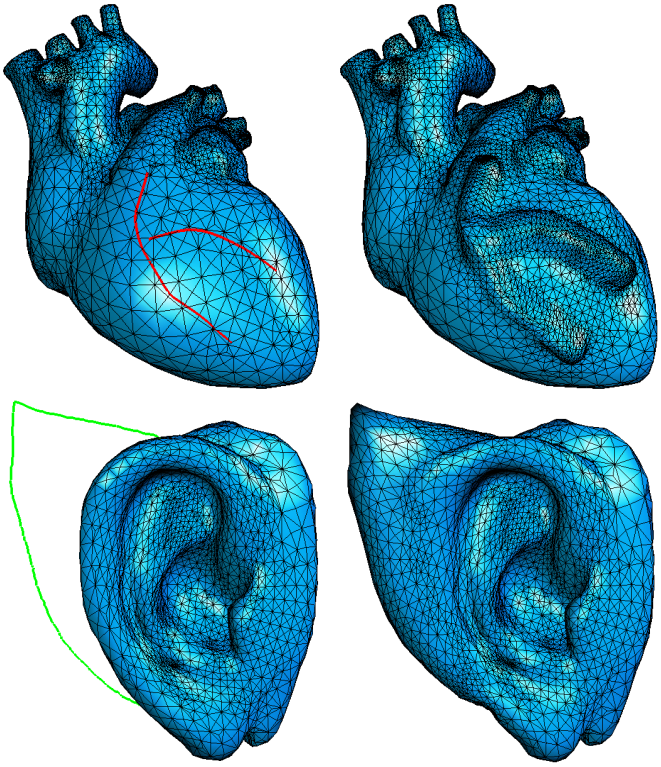


Fig. 2. Surface displacement: (top row) The vertices close to the sketches (red strokes) are displaced along the normal direction to produce the effect of arteries of the heart. (bottom row) The vertices in silhouette are displaced to new desired silhouette of a famous character (green stroke).

A. Drawing sketches

This sketching system uses a standard mouse device for input. Initially, the strokes are represented by an ordered sequenced of 2D points in window coordinates. To avoid irregular spacing between samples in a raw input stroke, we resample the input data on-the-fly by discarding any sample within a threshold distance of earlier samples. After the resampling, the input stroke is smoothed using *Chaikin subdivision scheme* [12].

In order to create the feature curves, the next step consists in transforming the input strokes from window coordinates to object/world coordinates by unprojecting each stroke sample into 3D space. This reverse projection can be done using the depth buffer and computing the inverse of modeling and viewing transformation matrix of the rendering API. In OpenGL API, the function `gluUnProject()` performs this task.

In our framework, the feature curve $\mathcal{C}$ is interpreted as a parametric curve of 3D points $\mathbf{p}_i$, i.e., $\mathcal{C} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$. The output of sketching stage is a set of feature curves generated through two processes: *surficial sketching* and *silhouette sketching*.

1) Surficial sketching: This process allows the graphical designers to sketch features on the surface of the model. After a sketch has been projected onto a surface, features are created by displacing the surface along the sketch. Usually the surface is displaced along the normal direction, suitable for creating details such as arteries and veins (see Figure 2.a).

2) Silhouette sketching: The main goal of this view-dependent sketch style is detecting the silhouette edges of the input surface and then sketch a new shape for this silhouette (see Figure 2.b). There are several strategies for silhouette detection in non-photorealistic rendering literature [13].

In order to compute a set of sample points $\mathcal{S}$ at the silhouette of a triangle mesh, we use object space silhouette edge detection algorithm proposed by Hertzmann and Zorin [14]. We select an edge $e = (\mathbf{v}_i, \mathbf{v}_j) \in E$ which satisfies:

$$\langle \mathbf{v}_i - \mathbf{c}, \mathbf{n}_i \rangle \langle \mathbf{v}_j - \mathbf{c}, \mathbf{n}_j \rangle \leq 0\,,$$

where $\mathbf{c}$ is the viewpoint, $\mathbf{n}_i$ and $\mathbf{n}_j$ are the vertex normals of $\mathbf{v}_i$ and $\mathbf{v}_j$, respectively. A silhouette point $\mathbf{p}^{sil}$ and its corresponding normal $\mathbf{n}^{sil}$ is computed by a linear interpolation:

$$\mathbf{p}^{sil} = (1-\alpha)\mathbf{p}_i + \alpha\mathbf{p}_j$$

$$\mathbf{n}^{sil} = (1-\alpha)\mathbf{n}_i + \alpha\mathbf{n}_j$$

with $\alpha = \langle \mathbf{v}_i - \mathbf{c}, \mathbf{n}_i \rangle \,/\, (\langle \mathbf{v}_i - \mathbf{c}, \mathbf{n}_i \rangle - \langle \mathbf{v}_j - \mathbf{c}, \mathbf{n}_j \rangle)$.

During silhouette sketch editing, the user first click the mouse button at a starting point $\mathbf{p}_0$ on the surface and drag the mouse to draw a new shape for the silhouette. To finish the drawing and create an end point $\mathbf{p}_n$, the user release the mouse button on the surface model. Then, we find the closest points $\mathbf{p}_0^{sil}$ and $\mathbf{p}_n^{sil}$ in $\mathcal{S}$ from $\mathbf{p}_0$ and $\mathbf{p}_n$, respectively. Finally, we project the silhouette sketch onto plane determined by the points $\mathbf{p}_0^{sil}$, $\mathbf{p}_n^{sil}$ and $\mathbf{p}_+^{sil} = 0.5(\mathbf{p}_0^{sil} + \mathbf{p}_n^{sil} + \mathbf{n}_0^{sil} + \mathbf{n}_n^{sil})$.

For our adaptive mesh refinement stage (Section III-B), it is important to note that the sketching stage must determine which face $f \in F$ the feature falls on surface, in other words, we must project the surficial and silhouette sketch onto input surface to produce a feature curve $\mathcal{C}$ on the surface. This projection can determined by an intersection test between ray and triangle [15], a very popular tool in ray-casting methods. The details of our multiresolution mesh structure will be discussed in the next section.

B. Adaptive Multiresolution Meshes

In order to perform the sketch-based modeling operations described in Subsection III-A using a representation oblivious strategy, as if we were working on a continuous deformable surface.

Recall that the above operations add shape features to the surface and, furthermore, notice that in order to create such features we need to deform the surface by a warping which is guided by the sketches in their regions of influence.

Since the surface in our system is represented by a triangle mesh, in order to accomplish our goal, we need to model a dynamic adapted mesh. To this end, we have chosen to adopt the framework introduced in [11], that applies the theory of stellar operators to create a variable resolution mesh representation.

For the sake of completeness we will review in this section the main concepts of variable resolution meshes and describe

how they are employed to build the adaptive dynamic mesh library used in the system.

A *multiresolution mesh* is a monotonic sequence of simplicial complexes $H = (M_0, M_1, \ldots, M_k)$, with increasing resolution, i.e., the mesh sizes, $|M_i| \leq |M_j|$, for $i < j$ and the meshes $M_i \in H$ are equivalent triangulations of a surface S. When a multiresolution mesh structure allows the resolution of the mesh to vary locally over the surface, it produces a *variable-resolution triangulation* [16]. The key to build mesh structures with this property is the ability to perform local refinement and simplification operations on the mesh. The theory of stellar subdivision provides such operators [17].

For the case of 2D meshes, the basic stellar operators are: the *face split* and its inverse *face weld*; and the *edge split* and its inverse *edge weld*. Variable resolution schemes modify a mesh locally using a sequence of these operations. These operations are illustrated in Figure 3.

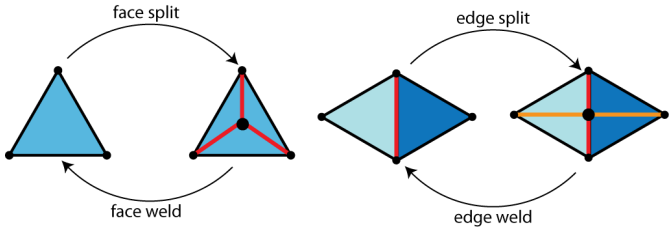


Fig. 3. Stellar operators on a mesh.

A powerful structure for variable resolution meshes is the *Regular Binary Multitriangulation* (RBM). The RBM is formed by applying edge splits to an initial coarse mesh M_0, called *base mesh*, that is a triangulated quadrangulation, or *tri-quad* mesh, such that this subjacent semi-regular subdivision structure is always maintained. In particular, the RBM is the mathematical abstraction used for the dynamic adapted mesh library in [11] and adopted in our system.

Given a sketch curve $\mathcal{C}$, we define a simple criterion for adaptive mesh refinement based on the distance between $\mathcal{C}$ and the vertices $\mathbf{v}_i \in V$ as follows:

$$\textbf{if} \quad \text{dist}\,(\mathbf{v}_i, \mathcal{C}) < r \quad \textbf{then} \quad \text{refine all triangles in } \mathbf{s}_i,$$

where r is a user threshold parameter and $\mathbf{s}_i$ is the vertex star of $\mathbf{v}_i$. Figure 4 illustrates this refinement process.

In our system, we generate the base mesh using first a mesh simplification algorithm, such as the Four-Face Cluster method of [18], and then imposing the tri-quad structure as described in [19]. The surface sampling can be done efficiently through a hierarchical parametrization which is built during the simplification process, similar to the method of [20]. Finally, the tests for refinement and simplification of the mesh are dependent of the surface adaptation to the geometry of the features created while the surface is deformed, as will be discussed in the next section.

C. Feature Generation

Feature generation is the most important stage in our mesh modeling framework. In this stage, the visible feature is actually generated by displacing vertices in the mesh. We divide feature generation into two steps. First step is called *vertex snapping*, this step consists in selecting which vertices will be displaced and create a geometry adjustment to approximate sharp features from these vertices. Second step is called *vertex displacement*, in this step; we displace the selected vertices to create a visually perceptible feature.

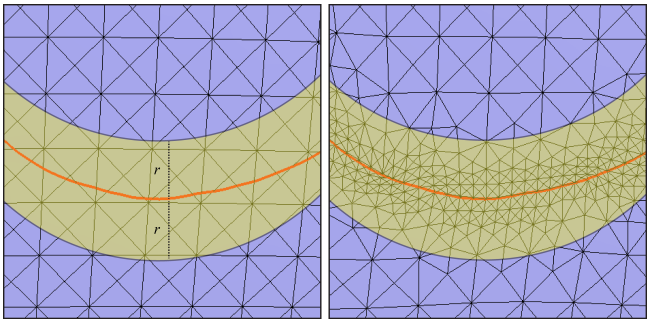


Fig. 4. Adaptive mesh refinement: (a) The sketch curve (red stroke) defines a mesh refinement region (yellow region). (b) The mesh adaptivity is performed in the refinement region.

1) Vertex snapping: The inputs to this stage of our framework are a multiresolution adaptive mesh from mesh refinement stage, and a set of feature curves. From these inputs, we determine a sequence of intersection points between edges $e \in E$ and each feature curve.

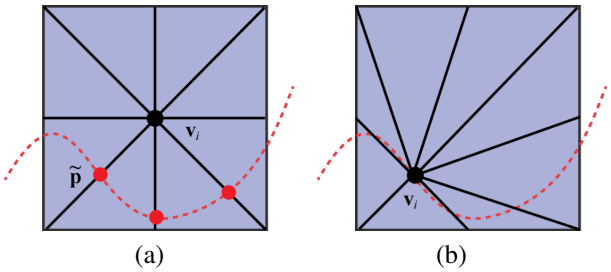


Fig. 5. Vertex snapping strategy: (a) Snap-to-curve uses the intersections points (red) between incident edges of a vertex $\mathbf{v}_i$ and the feature curve (dashed red curve). (b) The vertex $\mathbf{v}_i$ is moved to closest intersection point $\tilde{\mathbf{p}}$.

In order to create sharp features on surfaces, we adopt a vertex snapping strategy inspired in the *snap-to-curve* approach proposed by Biermann *et al.* [2]. First, our algorithm traverses each vertex $\mathbf{v}_i \in V$ taking its incident edges which have an intersection point with a feature curve. Then, we compute the closest intersection point $\tilde{\mathbf{p}}$ from $\mathbf{v}_i$. Finally, the vertex $\mathbf{v}_i$ is moved to $\tilde{\mathbf{p}}$. This snapping strategy is illustrated in Figure 5. Another advantage of our vertex snapping approach is to preserve the semi-regular structure of the mesh. This structural invariance allows us to increase the resolution of the mesh without recomputing the snapping (see Figure 6).

To alleviate the mesh distortion introduced in the vertex snapping process at the faces $f \in F$ surrounding the features. We use a vertex smoothing technique similar to the one used by Botsch and Kobbelt [21]: for each vertex $\mathbf{v}_i \in V$, the barycenter $\mathbf{b}_i$ of its star is computed. The vertex is then shifted

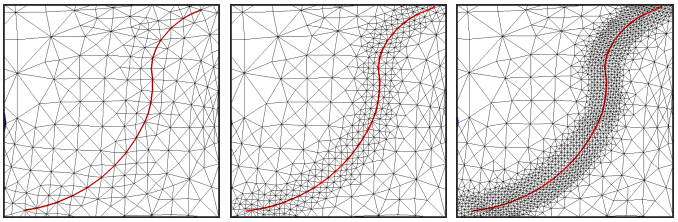


Fig. 6. Vertex snapping preserves the multiresolution structure of the mesh: the resolution increases in a neighborhood of the feature curve (red stroke) from coarse (left) to fine level (right).

tangentially towards $\mathbf{b}_i$, which is updated by:

$$\mathbf{v}_i = \mathbf{v}_i + (\mathbf{d}_i - \langle \mathbf{d}_i, \mathbf{n}_i \rangle \, \mathbf{n}_i) \ ,$$

where $\mathbf{d}_i = \mathbf{b}_i - \mathbf{v}_i$. The tangential movement avoids strong

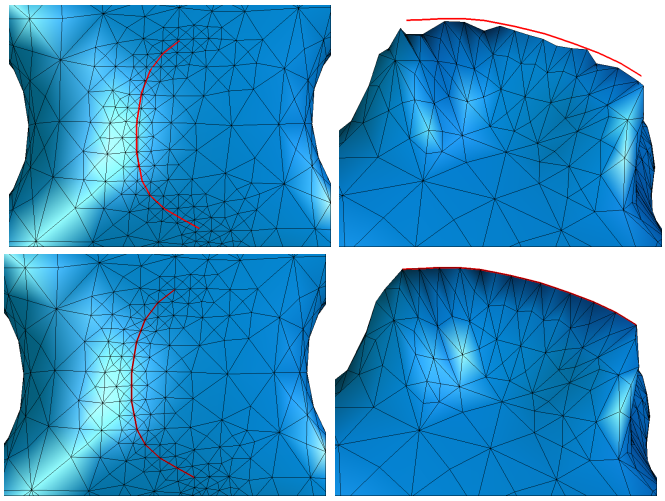


Fig. 7. Creation of a sharp feature combining vertex snapping and vertex displacement: Mesh and curve without vertex snapping (top). Mesh and curve with vertex snapping (bottom).

shrinking, and the relaxation towards $\mathbf{b}_i$ improves the aspect ratio of the triangles. Figure 7 demonstrates the result of this snapping strategy for a free-form sharp feature generation.

2) Vertex displacement: After the snapping stage, we create visually perceptible features moving the vertices along the sketch. In our system, the vertices can be displaced in two ways: along direction of the surface normal or along direction of the silhouette sketching.

To perform the displacement along the surface normal, we adopt the same approach proposed by Olsen *et al.* [3]. The user can control the displacement profile of each curve through a reduced set of intuitive scalar parameters, including width r, height h, and sharpness s. The width parameter r determines the radius of region of interest around a feature curve $\mathcal{C}$, i.e., the parameter r specifies the vertices to displace. The height h is a scaling parameter for the vertex normals along the feature.

The shape of a feature profile can be adjusted by the sharpness parameter $s \in [0, \infty)$, according to function $\Phi(\mathbf{x}) = \phi(\frac{dist(\mathbf{x},\mathcal{C})}{r})$, where

$$\phi(x) = \begin{cases} (1-x)^s & , \ x \in [0,1) \\ 0 & , \ \text{otherwise} \end{cases} \qquad (1)$$

The vertex $\mathbf{v}_i$ is moved to a new position by:

$$\mathbf{v}_i = \mathbf{v}_i + \Phi(\mathbf{v}_i)\mathbf{N}_i \ , \qquad (2)$$

where $\mathbf{N}_i = h\mathbf{n}_i$ denotes the scaled vertex normal direction. Figure 8 show the effect of sharpness parameter s on the feature profile.

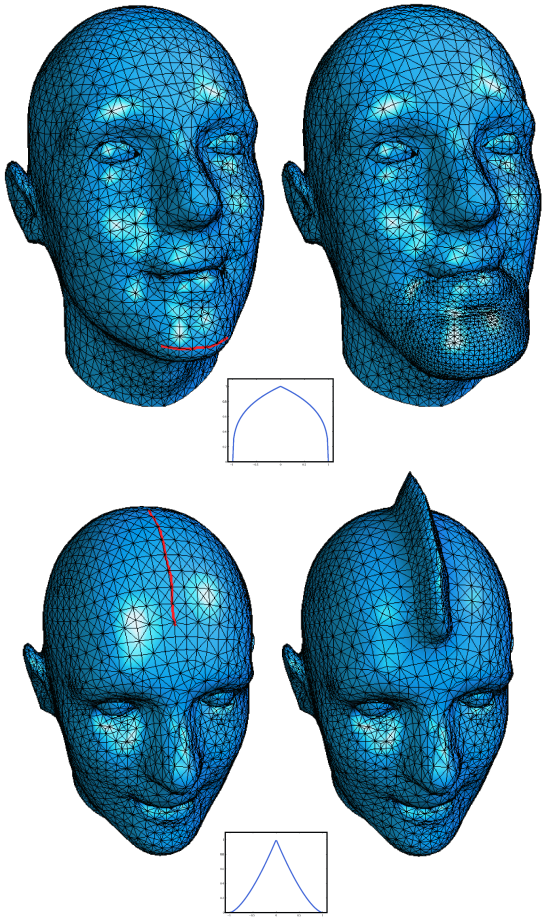


Fig. 8. The effect of varying the feature sharpness and the profiles of the displacement function (Equation 1): (top row) Original mesh (left); using $s = 0.3$ to create a smooth feature (right). (bottom row) Original mesh (left); using $s = 1.5$ to create a sharp feature (right).

The displacement along the silhouette sketching is computed replacing the scaled vertex normal $\mathbf{N}_i$ in Equation 2 by the direction $\bar{\mathbf{s}}_i = \mathbf{s}_i - \mathbf{p}_i$, where $\mathbf{p}_i \in \mathcal{C}$ is the closest point from a vertex $\mathbf{v}_i \in V$ and $\mathbf{s}_i$ is its corresponding point at the silhouette sketch.

IV. Results & Discussion

We experiment the modeling framework described in this work on various models. In particular, the augmentation effect on the mesh and the multiresolution adaptive structure of our system are clearly illustrated in Figure 9: (a) The user draws feature curves (red and yellow strokes) on a surface. (b) The refinement stage increases the mesh resolution around the features. (c) Then, the geometry of the mesh is modified to create surficial features. In our experiments, the generation of sharp features is a delicate task and the vertex snapping

process is necessary even in high resolution regions around the features as shown in Figure 9 (top row).

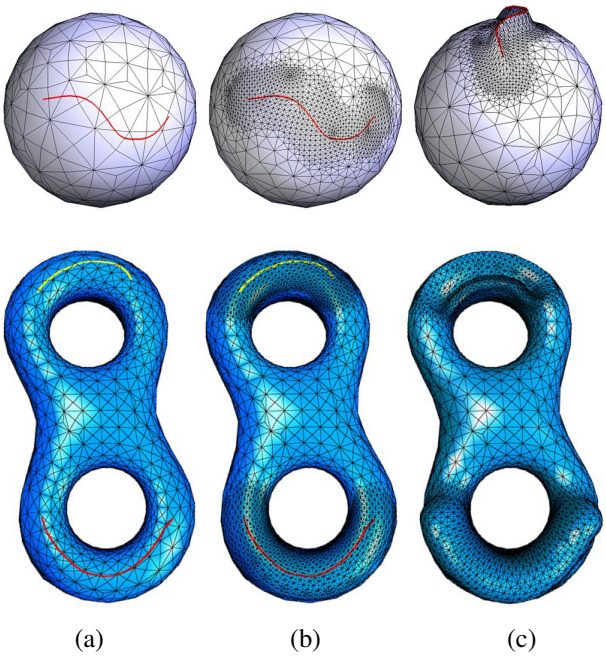


Fig. 9. Sketching on sphere (top row) and on a surface with complex topology, such as the bitorus surface (bottom row): (a) Sketching on original mesh; (b) Adaptive mesh refinement stage; (c) Feature generation stage.

The proposed framework provides a wide range of mesh augmentations just changing the sign of the height parameter h. Figures10.a demonstrates that negative values of h for the features (yellow strokes)leads an inward surface displacement, while Figure 10.b shows that positive values of h induce an outward surface displacement in a local vicinity of the feature curves (red strokes).

Figure 11 illustrates the results of the silhouette sketching of the proposed method. Figure 11.a shows how our system can be used to create medium-scale silhouette features on a handmade sculpture model. Figure 11.b presents the ability of our method to blend a silhouette sketch with the input mesh.

The proposed method can be used for terrain design which include hand-drawing directly on the terrain mesh to create hills, valleys and other features. We performed experiments over two terrain 4-8 meshes. In the first experiment, the input is a triangular mesh representing the terrain model. A 4-8 mesh is first generated directly from the given mesh (Figure 12). For a triangular mesh this conversion is accomplished by pairing triangles, the unpaired triangles are subdivided using a barycentric subdivision, adding 3 new triangles [22]. In the second experiment, the input is a height-map of the terrain model. Figure 13 illustrates the process. A standard 3D planar mesh grid is first generated, with constant X,Y spacing and Z modulated after the grayscale intensities of the height-map image. This results in a regular quad base mesh which is ideal to be converted in a 4-8 mesh. The quad mesh triangulation is straightforward and generates triangles with a good aspect ratio. The semi-regular structure is essential in the refining process since the stellar operators rely on this regularity. Figure 14 shows results for another height-map and 3 strokes for the 4-8 mesh augmentation. Note that the base mesh in both examples are a coarse approximation to the original height-map image. The user can adjust the resolution of the base mesh for experimenting with both coarse and fine approximations. This approach is useful when the terrain design requirement is to capture the overall shape of a given terrain, and allowing the user to augment new shape features.

Results presented in this paper were generated on a 1.86GHz Centrino with 2GB of RAM. Table I describes some usage statistics recorded while producing the results. The reported timings include the time t_{ref} spent to refine the mesh, the time t_{snap} spent snapping the vertices, and the time t_{feat} spent applying the features by the system, as well as the number of faces of the input model $\#\triangle_{in}$ and the number of faces of the augmented model $\#\triangle_{out}$. The entire time required to augment a mesh depends on both the number and complexity of feature curves, as well as the complexity and the resolution of the input mesh. Based on our experiments and observations, the potential bottleneck in our approach is due to the vertex snapping stage.

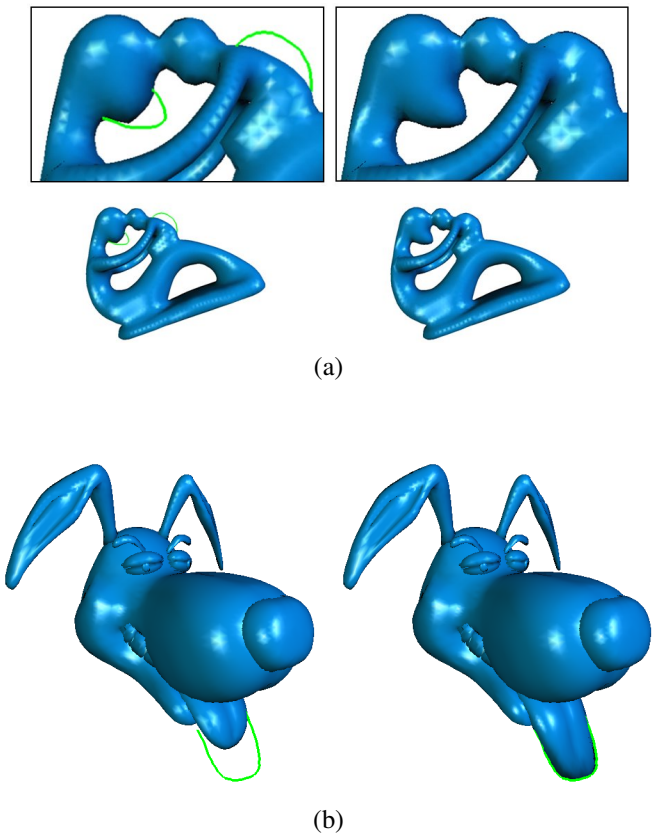
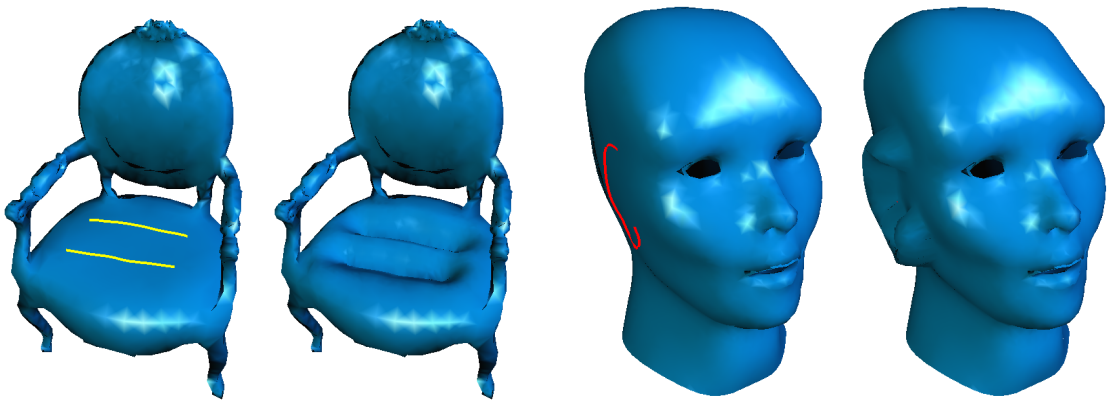


Fig. 11. Silhouette sketching: (a) The Fertility model (left) is modified with medium-scale silhouette features (right). (b) The tongue of a stylized dog head model (left) can be augmented according to silhouette feature generated by our method (right).



(a) (b)

Fig. 10. Surficial displacement: (a) Inward surface displacement, features curves (yellow strokes) are placed onto Victorian chair model (left) to create a more comfortable appearance (right); (b) Outward surface displacement, the alien head mesh (left) is augmented to produce an ear structure (right).

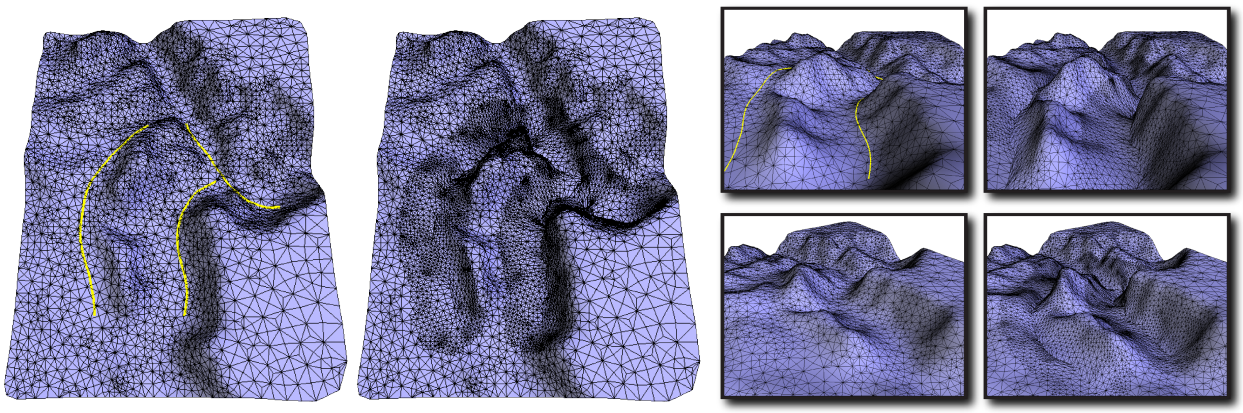


Fig. 12. Different views of before and after augmenting a 4-8 mesh generated directly from a given triangular mesh of the Grand Canyon model.

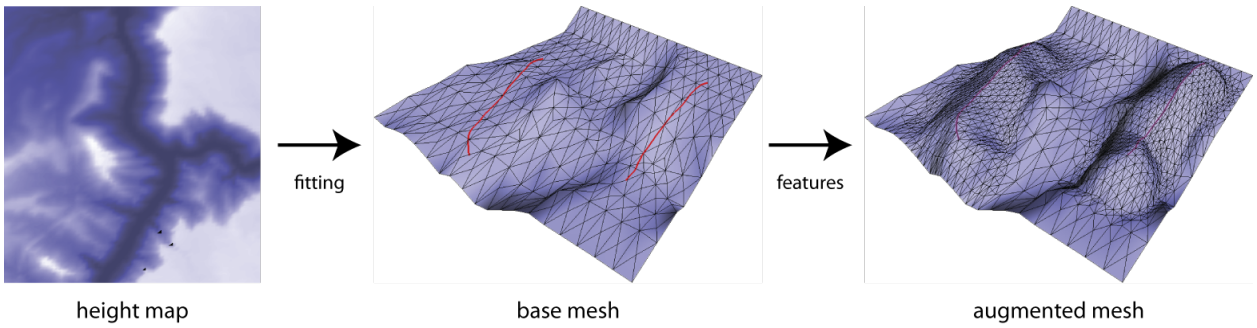


Fig. 13. Sketch-based augmentation pipeline over a terrain model generated from a height-map of the Grand Canyon.

V. Conclusion & Future Work

In this paper, we have presented a novel modeling framework for augmenting an input mesh model using different sketch-based techniques with multiscale control of the sharp features. Our method combines the sketching process into a hierarchical adaptive mesh refinement process based on stellar theory. We do not require any kind of special subdivision rules to further enhance the augmented mesh; and finally, the adaptive mesh refinement allows the mesh resolution to increase only in the vicinity of the feature curves.

In terms of future work, one natural extension of the techniques introduced in this paper is to use the curves created by the user in the sketch based deformation as "first class" elements in the modeling system. That implies in considering these curves as *handles*, which define, parametrize and control features of the model. In this case, it would be instrumental to represent such handles as subdivision curves that lie on the surface and can be edited as a free-form element. One way to implement functionality is through geodesic subdivision as proposed by Morera *et al.* [23].

Silhouette detection in our system is made in object space. However, the silhouette path on the mesh might fold onto itself when projected to image space. To avoid this problem and improve the performance of our system, we can explore the

TABLE I
SOME STATISTICS AND TIMINGS (IN SECONDS) OF OUR SYSTEM.

model	*fig*	$\#\triangle_{in}$	$\#\triangle_{out}$	t_{ref}	t_{snap}	t_{feat}
Bull	*1*	19k	21k	0.06	1.85	0.02
Heart	*2*	15k	19k	0.09	1.64	0.06
Ear	*2*	5k	8k	0.06	1.13	0.08
Head I	*8*	9k	16k	0.09	0.72	0.09
Head II	*8*	9k	11k	0.06	0.69	0.06
Sphere	*9*	0.7k	3k	0.01	0.17	0.02
Bitorus	*9*	3k	7k	0.05	1.21	0.04
Chair	*10.a*	6k	8k	0.05	1.32	0.03
Alien	*10.b*	7k	9k	0.04	0.74	0.03
Fertility	*11.a*	13k	17k	0.16	2.11	0.09
Dog	*11.b*	28k	35k	0.37	3.51	0.22
Canyon	*12*	9k	19k	0.18	2.81	0.08
Canyon	*13,*	0.7k	3.2k	0.02	0.34	0.01
Moonscape	*14*	0.7k	5.2k	0.03	0.43	0.03

image space silhouette detection algorithms as described by Zimmermann *et al.* [6].

Another extension would be the creation of other sketch operators such as brush strokes for mesh smoothing or flattening.

Finally, our sketching interface could be improved. Rather than having user-tunable parameters such as height and sharpness, we can predefine wide range of expressive strokes, where each stroke would represent some preset combination of feature parameters.

ACKNOWLEDGMENT

We would like to thank Emilio Vital Brazil for his useful discussions and advice. We also thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported by grants from the Brazilian funding agencies CNPq, FAPESP, INCT-MACC, and by the Alberta Innovates Academy (AITF) / Foundation CMG Industrial Research Chair in Scalable Reservoir Visualization.

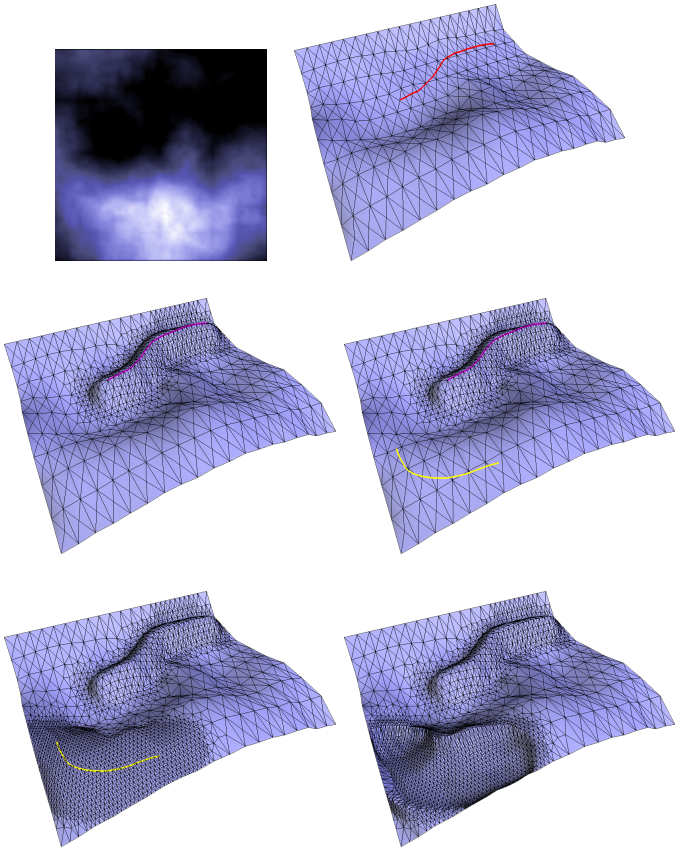


Fig. 14. Sketch-based augmentation over a terrain model generated from a moonscape heigh-tmap. Top row, the input height-map and the resulting 4-8 terrain mesh and a first stroke input. Middle row,the resulting augmentation and the input of a second stroke. Bottom row, sketching a third stroke over the augmented regions.

REFERENCES

[1] L. Olsen, F. Samavati, M. Sousa, and J. Jorge, "Sketch-based modeling: A survey," *Computer & Graphics*, vol. 33, no. 1, pp. 85–103, 2009.
[2] H. Biermann, I. M. Martin, D. Zorin, and F. Bernardini, "Sharp features on multiresolution subdivision surfaces," in *9th Pacific Conference on Computer Graphics and Applications*, 2001, pp. 140–149.
[3] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based mesh augmentation," in *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2005, pp. 43–52.
[4] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge, "Shapeshop: Sketch-based solid modeling with blobtrees," in *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2005, pp. 53–62.
[5] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1142–1147, 2005.
[6] J. Zimmermann, A. Nealen, and M. Alexa, "Sketching contours," *Computers & Graphics*, vol. 32, no. 5, pp. 486–499, 2008.
[7] A. Khodakovsky and P. Schröder, "Fine level feature editing for subdivision surfaces," in *SMA '99: Proc. of the fifth ACM symposium on Solid modeling and applications*, 1999, pp. 203–211.
[8] Y. Gingold and D. Zorin, "Shading-based surface editing," *ACM Trans. Graph.*, vol. 27, pp. 95:1–95:9, August 2008.
[9] B. R. De Araujo and J. A. P. Jorge, "A calligraphic interface for interactive free-form modeling with large datasets," in *Proc. of the 18th Brazilian Symposium on Computer Graphics and Image Processing*, 2005, pp. 333–340.
[10] K. Takayama, R. Schmidt, K. Singh, T. Igarashi, T. Boubekeur, and O. Sorkine, "Geobrush: Interactive mesh geometry cloning," *Computer Graphics Forum*, vol. 30, no. 2, pp. 613–622, 2011.
[11] L. Velho, "A dynamic adaptive mesh library based on stellar operators," *Journal of Graphics Tools*, vol. 9, no. 2, pp. 1–29, 2004.
[12] G. Farin, *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.
[13] A. Hertzmann, "Introduction to 3d non-photorealistic rendering: Silhouettes and outlines," in *SIGGRAPH Course Notes*, 1999.
[14] A. Hertzmann and D. Zorin, "Illustrating smooth surfaces," in *Proc. of SIGGRAPH 2000*, 2000, pp. 517–526.
[15] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *Journal of Graphics Tools*, vol. 2, no. 1, pp. 21–28, 1997.
[16] L. Velho and J. Gomes, "Variable resolution 4-k meshes: Concepts and applications," *Computer Graphics Forum*, vol. 19, no. 4, pp. 195–214, 2000.
[17] T. Lewiner, H. Lopes, E. Medeiros, G. Tavares, and L. Velho, "Topological mesh operators," *Computer Aided Geometric Design*, vol. 27, no. 1, pp. 1–22, 2010.
[18] L. Velho, "Mesh simplification using four-face clusters," in *Proc. of SMI 2001*, 2001.
[19] L. Velho and D. Zorin, "4-8 subdivision," *Computer-Aided Geometric Design*, vol. 18, no. 5, pp. 397–427, 2001.
[20] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," in *Proc. of SIGGRAPH 98*, 1998, pp. 95–104.
[21] M. Botsch and L. Kobbelt, "A remeshing approach to multiresolution modeling," in *Symposium on Geometry Processing*, 2004, pp. 185–192.
[22] L. Velho, "Semi-regular 4-8 refinement and box spline surfaces," in *Proc. of the 13th Brazilian Symposium on Computer Graphics and Image Processing*, 2000, pp. 131–138.
[23] D. M. Morera, P. C. Carvalho, and L. Velho, "Modeling on triangulations with geodesic curves," *The Visual Computer*, vol. 24, no. 12, pp. 1025–1037, 2008.