To appear in the ACM SIGGRAPH conference proceedings

A Sketch-Based Interface for Detail-Preserving Mesh Editing

Andrew Nealen, TU Darmstadt
Olga Sorkine, Tel Aviv University
Marc Alexa, TU Darmstadt
Daniel Cohen-Or, Tel Aviv University

Abstract

In this paper we present a method for the intuitive editing of surface meshes by means of view-dependent sketching. In most existing shape deformation work, editing is carried out by selecting and moving a *handle*, usually a set of vertices. Our system lets the user easily determine the handle, either by silhouette selection and cropping, or by sketching directly onto the surface. Subsequently, an edit is carried out by sketching a new, view-dependent handle position or by indirectly influencing differential properties along the sketch. Combined, these editing and handle metaphors greatly simplify otherwise complex shape modeling tasks.

Keywords: Sketch Based Model Editing, Laplacian Surface Editing, Differential Geometry, Sketching, Deformations

1 Introduction

A few strokes suffice to sketch the main features of a shape. This is why designers still prefer using pen and paper to invent and communicate, and explains the great success of sketch-based shape modeling approaches, such as SKETCH [Zeleznik et al. 1996] and Teddy [Igarashi et al. 1999]. In this work, we add to the existing toolbox of sketch based shape modeling techniques. Our contribution is a tool for sketching significant shape details on already existing coarse or detailed shapes. We believe the important first step of creating the basic shape from scratch is essentially solved: either based on sketching (apart from the pioneering works mentioned above, see also [Karpenko et al. 2002; Igarashi and Hughes 2003; Bourguignon et al. 2004]) or using other modeling techniques. Ideally, a sketch-based modeling system for 3D shapes should use the very same sketches that designers would draw on a piece of paper to convey the shape. What are these lines? As pointed out by Hoffman and Singh [1997], the human visual system uses silhouettes as the first index into its memory of shapes, making everyday objects recognizable without color, shading or texture, but solely by their contours. In the area of non-photorealistic rendering (NPR), silhouettes have been used extensively [Gooch and Gooch 2001] and recently they have been extended to *suggestive* contours: curves on the shape that might be silhouettes in nearby views [DeCarlo et al. 2003]. The apparent presence of a feature line in a picture of a shape results from an abrupt change in illumination. Apart from view dependent features for which this happens or might happen in a nearby view, change of illumination generally correlates with curvature. Lines of curvature extrema (i.e. ridges and ravines) have, therefore, also been used in NPR for conveying shape.

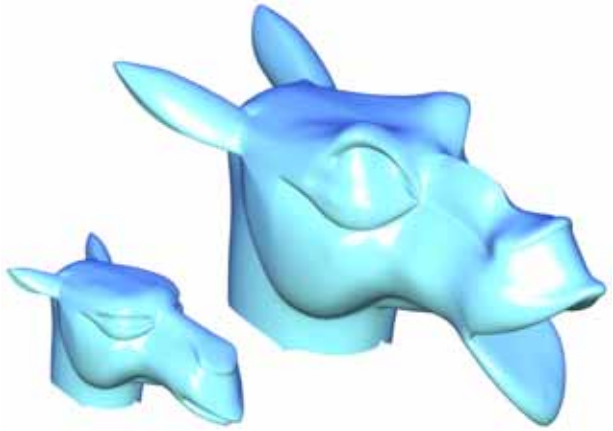

Figure 1: With a few strokes we have greatly increased the expressiveness of the CAMEL model (bottom left). See Fig. 2 for details.

We come to the conclusion that **sketching** a shape is **inverse NPR**. Consequently, we design a sketch-based modeling interface using silhouettes and sketches as input, and producing contours, or suggestive contours, and ridges/ravines. The user can sketch a curve, and the system adapts the shape so that the sketch becomes a feature line on the model, while preserving global and local geometry as much as possible. As the requested properties of the sketch cannot or should not always be accommodated exactly, users only *suggest* feature lines.

It might seem obvious to let users alter contours, or ask for a line in space to be a feature line. Interestingly, our concurrent goals of preserving the global and local geometry during the edit while using feature lines for defining the edit are difficult to implement using traditional approaches: typical sketching tools [Igarashi et al. 1999; Karpenko et al. 2002; Fleisch et al. 2004] do use silhouettes, however, they create only smooth shapes. Some operations of sketching techniques might preserve geometric detail, however, they are not based on inserting feature lines into the shape [Draper and Egbert 2003; Kho and Garland 2005]. In general modeling environments, such as space deformation techniques (e.g., [Sederberg and Parry 1986; Singh and Fiume 1998]) and multi-resolution or subdivision mesh modeling approaches [Zorin et al. 1997; Kobbelt et al. 1998; Biermann et al. 2000], it can be difficult to incorporate the displacement of a feature curve: these approaches provide a basis that spans a space of shapes; the requested displacement has to be translated into coefficients of this basis. In general, this might be impossible, and an approximate solution typically leads to a difficult inverse problem (see also Botsch and Kobbelt [2004]). Our idea becomes realizable through the recent trend to cast mesh modeling problems as discrete Laplace or Poisson models [Alexa 2003; Botsch and Kobbelt 2004; Sorkine et al. 2004; Yu et al. 2004; Sumner and Popović 2004]. Within this framework, it is easy to displace a set of edges (e.g., sketch a new position of an identified contour) while preserving the geometric details of the surface as much as possible. However, most of the feature lines we want to use have specific differential properties, either absolute or relative to the viewing direction, and they might not coincide with edges on the mesh. We therefore extend the framework of Laplace/Poisson mesh modeling in the following ways: (a) we accommodate constraints on the normals and the curvature; (b) we allow constraints to be placed

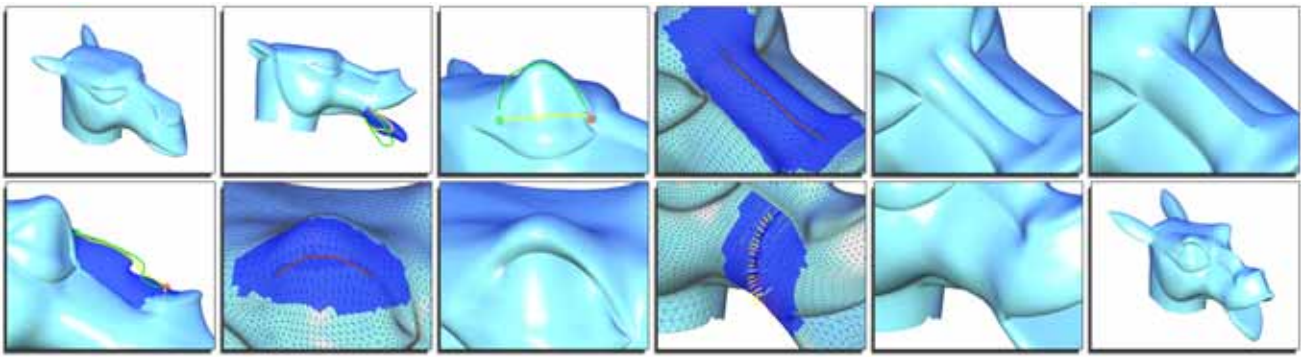

Figure 2: Our mesh editing tool in action. Top row [(1)-(6)]: First, we open the mouth of the CAMEL model (1) by detecting an object silhouette, and sketching an approximation of the lip shape we want (2) (See Section 3). Note that in (2) the yellow curve is the original object silhouette, the green curve is the user sketch, and the dark blue region is the result of a previously placed sketch. By sketching directly onto the model (3) we produce a handle (yellow) by which we can lift the eyebrow with the green sketch. For the creation of sharp features we sketch the feature line (4) and then scale the affected Laplacians to produce either a ravine (5) or a ridge (6) (See Section 4.2). Bottom row [(7)-(12)]: If we are not yet satisfied with the ridge in (6), we can edit the newly created object contour using our silhouette tool (7). Sketching a ravine under the eye by geometry adjustment (See Section 4.1) is shown in (8) and (9). Finally, we sketch a subtle suggestive contour near the corner of the mouth in (10) and (11) (See Section 4.3), resulting in the SCREAMING CAMEL model (12), shown in Fig. 1.

on virtual vertices, i.e. vertices placed on edges that only serve to implement the constraints but are never added to the mesh; (c) we incorporate a tangential mesh regularization, which moves edges onto sharp features while ensuring well-shaped triangles.

This mesh modeling framework together with a user-interface mostly based on sketching suggested feature lines onto or around a shape, indeed, yields an intuitive shape modeling technique.

2 Mesh modeling framework

The basic idea of the modeling framework is to satisfy linear modeling constraints (exactly, or in the least squares sense), while preserving differential properties of the original geometry. This technique has recently been presented in various fashions and we only briefly explain the main concepts. For more detailed explanations see the references given below. One way of deriving these linear constraints is to ask that the Laplacian of the original geometry be preserved in the least squares sense [Alexa 2003; Lipman et al. 2004]. Let the mesh be represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, consisting of vertices $\mathbf{V}$ and edges $\mathbf{E}$. Let $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$, $\mathbf{v}_i = (v_{i_x}, v_{i_y}, v_{i_z}) \in \mathbb{R}^3$ be the original geometry and Δ the Laplace operator, then the deformed geometry $\mathbf{V}'$ is defined by the constrained minimization

$$\mathbf{V}' = \arg\min_{\mathbf{W}} \|\Delta\mathbf{V} - \Delta\mathbf{W}\|^2, \tag{1}$$

where the vertices might be weighted differently to trade-off between modeling constraints and the reproduction of original surface geometry. Note that this is equivalent to solving a linear system of the form $\mathbf{AV}' = \mathbf{b}$ in the least squares sense. If the original surface was a membrane, the necessary constraints for the minimizer lead to $\Delta^2\mathbf{V}' = 0$, which has been advocated by Botsch and Kobbelt [2004] in the context of modeling smooth surfaces. If, in contrast, the original surface contained some detail, the right-hand side is non-zero and we arrive at a variant of the discrete Poisson modeling approach of Yu et al. [2004].

The modeling operation is typically localized on a part of the mesh. This part of the mesh is selected by the user as the region of interest (ROI) during the interactive modeling session (with a lasso tool). The operations are restricted to this ROI, padded by several layers of anchor vertices. The anchor vertices yield positional constraints $\mathbf{v}'_i = \mathbf{v}_i$ in the system matrix $\mathbf{A}$, which ensure a gentle transition between the altered ROI and the fixed part of the mesh. Based on the constraints formulated so far, local surface detail is preserved if parts of the surface are translated, but changes with rotations and scales. One way of dealing with this is to define local rotations per vertex a priori. Lipman et al. [2004] compute these rotations from a smoothed solution of Eq. 1, Yu et al. [2004] let the user specify a few constraint transformations and then interpolate them over the surface. However, we would like to incorporate the treatment of directions into the modeling phase so that some of the details have a fixed (normal) orientation, while others may rotate. Thus, we adopt the approach of Sorkine et al. [2004], who define the local rotations and scales by comparing one-rings between $\mathbf{V}$ and $\mathbf{V}'$. However, we discretize the Laplace operator using cotangent weights as recommended by Meyer et al. [2003]. The conditions to be satisfied lead to an overdetermined system of linear equations of the form $\mathbf{AV}' = \mathbf{b}$, which we solve in the least squares sense according to the normal equations $\mathbf{A}^T\mathbf{AV}' = \mathbf{A}^T\mathbf{b}$. For information on how to derive the rows resulting from Eq. 1 see [Sorkine et al. 2004].

We extend this framework towards constraints on arbitrary points on the mesh. Note that each point on the surface is the linear combination of two or three vertices. A point on an edge between vertices i and j is defined by one parameter as $(1-\lambda)\mathbf{v}_i + \lambda\mathbf{v}_j$, $0 \le \lambda \le 1$. Similarly, a point on a triangle is defined by two parameters. We can put positional constraints $\hat{\mathbf{v}}_{ij}$ on such a point by adding rows to the system matrix $\mathbf{A}$ of the form

$$(1-\lambda)v'_{i_x} + \lambda v'_{j_x} = \hat{v}_{ij_x}, \ldots . \tag{2}$$

Furthermore, we extend the framework by using other forms of differentials to achieve some additional effects. Let δ_i be the Laplacian of $\mathbf{v}_i$, the result of applying the discrete Laplace operator to $\mathbf{v}_i$, i.e.

$$\delta_i = \mathbf{v}_i - \sum_{\{i,j\}\in E} w_{ij}\mathbf{v}_j, \tag{3}$$

where $\sum_{\{i,j\}\in E} w_{ij} = 1$, and the weights w_{ij} are determined using the cotangent weights [Meyer et al. 2003]. An important benefit of this weighting is that δ_i points in the normal direction, and the length $\|\delta_i\|$ is proportional to the mean curvature around vertex i. This allows us to prescribe a certain normal direction and/or curvature for a vertex, simply by adding a row to $\mathbf{A}$ of the form

$$\mathbf{v}'_i - \sum_{\{i,j\}\in E} w_{ij}\mathbf{v}'_j = \delta'_i. \tag{4}$$

Setting the normal direction is necessary for contours and suggestive contours, setting the curvature – for ridges or ravines.

To access the tangential location of vertices, we use the umbrella operator [Kobbelt et al. 1998] as a discrete Laplacian and relate it to the cotangent weighted Laplace operator. We exploit this for regularizing the mesh in tangential direction, by asking that

$$\mathbf{v}'_i - d_i^{-1} \sum_{\{i,j\}\in E} \mathbf{v}'_j = \mathbf{v}_i - \sum_{\{i,j\}\in E} w_{ij}\mathbf{v}_j, \qquad (5)$$

where d_i is the degree of vertex i. The rationale behind this operation is this: the uniformly weighted operator generates a tangential component, while the cotangent weighting does not. Asking that they are equivalent is essentially solving the Laplace equation but only for the tangential components. The result is a mesh with well shaped triangles, preserving the original mean curvatures as long as the tangential offset is not too large. Note that we typically restrict this operation to small regions, so that large tangential drift cannot occur.

In the following sections, we explain how to use these basic building blocks for satisfying user-defined feature lines on a mesh.

3 Silhouette sketching

Our goal is to identify areas of the model which are easily recognized, and for which our memories hold vast databases of possible variations, and then apply these variations by sketching them. The idea is simple yet effective: after defining a region of interest on the surface and a camera viewpoint, we select (and trim) one of the resulting silhouettes, and then sketch a new shape for this silhouette (see Fig. 3).

For the computation of silhouettes on polygonal meshes, various methods are available, see [Hertzmann 1999]. We have chosen to use object space silhouettes, and include the ability to switch between edge silhouettes (mesh edges, for which one adjacent face is front-facing and one is back-facing) and smooth surface silhouettes [Hertzmann and Zorin 2000]. Hertzmann and Zorin [2000] determine the silhouette on mesh edges $e = (\mathbf{v}_i, \mathbf{v}_j)$ by linearly interpolating corresponding vertex normals $\mathbf{n}_i, \mathbf{n}_j$: a silhouette point $\mathbf{p} = (1-\lambda)\mathbf{v}_i + \lambda\mathbf{v}_j$ on e has to satisfy $((1-\lambda)\mathbf{n}_i + \lambda\mathbf{n}_j)\cdot(\mathbf{p}-\mathbf{c}) = 0$, where $\mathbf{c}$ is the viewpoint. Silhouette points on edges are connected by segments over faces.

During editing, the user first picks one of the connected components, and then interactively adjusts the start and end point by dragging them with the mouse. Note that degenerate silhouette edge paths might lead to multiply connected curves, resulting in non-intuitive user interaction. Smooth silhouettes [Hertzmann 1999] remedy this problem on smoothly varying surfaces, and only for models with distinct sharp features (such as CAD models), mesh edges are used as silhouettes. In any case, the selected silhouette segment is represented as a set of points $\mathbf{q}_i$ on the mesh.

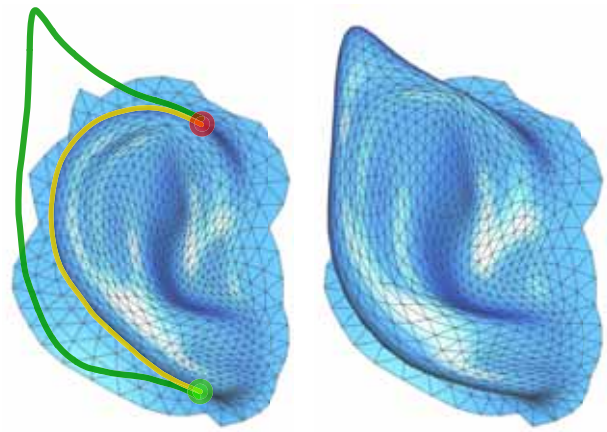

Figure 3: Sketching a very recognizable ear silhouette: we detect, select, crop and parameterize an object silhouette (yellow, the green and red balls represent begin and end vertices respectively), and then sketch a new desired silhouette (green).

After selecting a silhouette segment, the user sketches a curve on the screen, representing the suggested new silhouette segment. The sketch is represented as a polyline in screen space. The vertex locations $\mathbf{s}_i$ on this polyline result in constraints on mesh vertices as follows: First, silhouette vertices $\mathbf{q}_i$ are transformed to screen space, i.e. the first two components contain screen space coordinates, while the third contains the z-value. Then, both curves are parameterized over $[0,1]$ based on edge lengths of the screen space polylines. This induces a mapping from $\mathbf{q}_i$ to $\{\mathbf{s}_j\}$, defining a new screen space position $\mathbf{q}'_i$ (note that $\mathbf{q}'_i$ retains the z-value of $\mathbf{q}_i$).

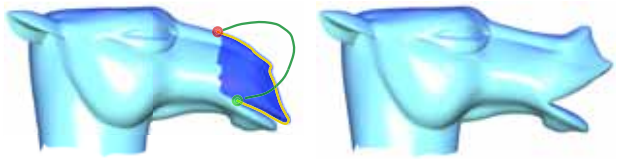

Figure 4: Sketching an *approximate* CAMEL lip by reducing the weights on the positional constraints for silhouette vertices.

The new position $\mathbf{q}'_i$ in screen space is transformed back to model space and serves as a positional constraint. Note that when using smooth surface silhouettes, on-edge constraints have to be used (see Eq. 2). Additionally, varying the weighting of positional constraints along the silhouette against Laplacian constraints leads to a trade off between the accurate positioning of silhouette vertices under the sketch curve, and the preservation of surface details in the ROI. To achieve this, we simply multiply the affected rows in $\mathbf{A}$ and $\mathbf{b}$ with the selected weighting factor. For example, the result in Fig. 3 follows the sketch closely, whereas the sketch in Fig. 4 only hints at the desired lip position.

This method works well even for moderately noisy and bumpy surfaces and preserves details nicely (see Fig. 5). Note that for very noisy surfaces, object space silhouette paths and loops may become arbitrarily segmented, in which case our silhouette sketching method is no longer applicable. In such cases, sketch editing can be performed relative to any user-defined curve sketched manually onto the surface, as was done for lifting the eyebrows of the CAMEL, see Fig. 2(3).

The matrix $\mathbf{A}^T\mathbf{A}$ is computed and factored once for each ROI and silhouette curve selection, and we simply solve for each sketch by back substitution [Toledo 2003]. Some editing results in Fig. 1 were obtained by using the silhouette editing capabilities of our system: sketching larger ears, opening the mouth and modifying the nose contour.

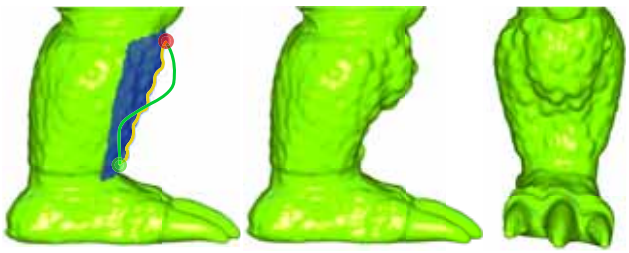

Figure 5: Editing the bumpy ARMADILLO leg: although the silhouette (yellow) in the ROI (blue) has substantial depth variation and the desired silhouette (green) is smooth, properly weighting the positional constraints retains the surface characteristics after the edit.

4 Feature and contour sketching

4.1 Geometry adjustment

Suppose we intend to create a potentially sharp feature where we have drawn our sketch *onto* the mesh. To create a meaningful feature (i.e. a ridge, ravine or crease) on a mesh, we must first adjust the mesh geometry to accommodate such a feature directly under the sketch, since in our setting the sketch need not run along an edge path of the mesh. To illustrate this, see Fig. 6(a), where the sketch path $\{\mathbf{s}_i\}$ (green) follows the edges on the left, but runs perpendicular to them on the right. By applying repeated subdivision we could have locally adjusted the mesh resolution, but for situations similar to the one in Fig. 6(a), many levels of subdivision would be necessary to properly approximate the sketch with an edge path. Another option would be to cut the mesh along the sketch; however, we have found a simpler method that avoids increasing the mesh complexity, yields nice feature lines and well-shaped triangles while retaining the original mesh topology. In detail:

The triangles in the ROI are transformed to screen space; triangles intersecting $\{\mathbf{s}_i\}$ are gathered (Fig. 6(a), dark triangles) and the begin and end mesh vertices are identified.

An edge path $\mathbf{V}_p = (\mathbf{v}_{p_1}, \mathbf{v}_{p_2}, \ldots, \mathbf{v}_{p_n})$ that is close to $\{\mathbf{s}_i\}$ is computed by solving a weighted shortest path problem in the edge graph of the ROI. The weight for each edge is the sum of its vertices' screen space distance to $\{\mathbf{s}_i\}$. The resulting edge path vertices are generally not on, but close to $\{\mathbf{s}_i\}$ (shown in red in Fig. 6(a)).

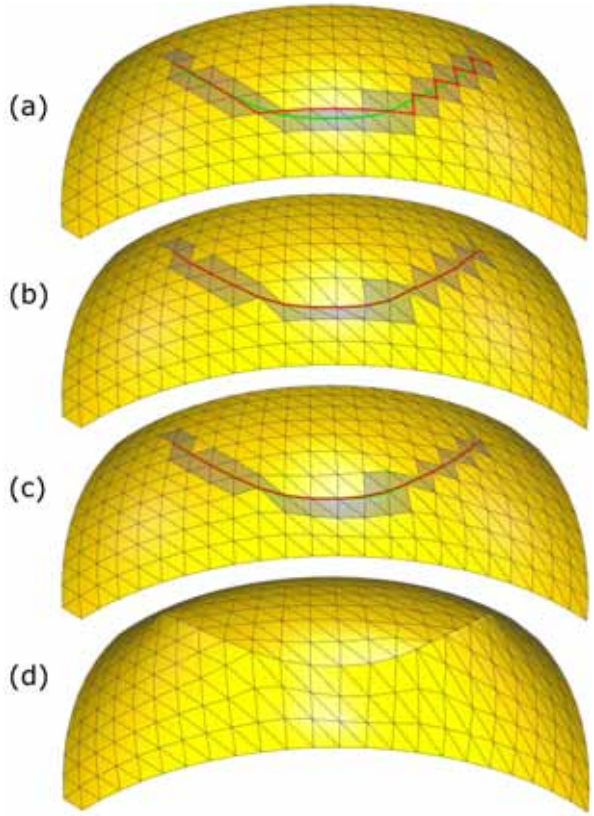

Figure 6: Creating a ravine-like crease: in (a) the green sketch given by the user is approximated by the red edge path on the original geometry. We adjust the geometry to lie directly under the sketch by orthogonal projection along the tangent plane (b), and then relax the area around the sketch (c). Now we can create the crease by scaling the Laplacians along the edge path (d), resulting in a sharp feature, even for this coarsely sampled surface.

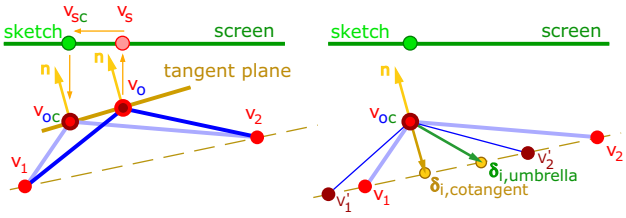

Figure 7: Adjusting edge path vertices to lie under the sketch curve (left): an object-space edge path vertex $\mathbf{v}_o$ is projected to $\mathbf{v}_s$ in screen space, from there orthogonally projected onto $\mathbf{v}_{sc}$ on the sketch curve, and then projected back onto the tangent plane defined by the normal at $\mathbf{v}_o$, yielding the new vertex position $\mathbf{v}_{oc}$. Relaxing the sketch region (right): to ensure a good triangulation after adjusting the geometry, we perform a relaxation of the edge-path vertices (allow them to move along the sketch path) and nearby vertices by constraining $\delta_{i,umbrella}$ to $\delta_{i,cotangent}$ in the least squares sense. Qualitatively, this moves $\mathbf{v}_1$ and $\mathbf{v}_2$ to $\mathbf{v}'_1$ and $\mathbf{v}'_2$, while keeping $\mathbf{v}_{oc}$ under the sketch.

The path vertices $\mathbf{V}_p$ are mapped onto closest edges of the sketch path $\{\mathbf{s}_i\}$ in screen space; corresponding z-values are computed from restricting each vertex to move on its tangent plane, as defined by the original vertex normal (Fig. 7, left). The resulting edge path closely follows the sketch curve (Fig. 6(b)), yet may introduce badly shaped triangles.

We improve triangle shapes by relaxing vertices close to the sketch so that their umbrella Laplacian equals the cotangent Laplacian in the least squares sense (See Fig. 7, right, and Section 2). For the vertex relaxation we must solve a linear system, much like the actual editing solver, but with constraints given by Eq. 5. Obviously, the edge path vertices must remain under the sketch path during this procedure. To ensure this, while also giving the edge path vertices a valid degree of freedom, we add them as positional constraints (Section 2), and additionally add averaging constraints of the form

$$\mathbf{v}'_{p_i} - \frac{1}{2}\mathbf{v}'_{p_{i-1}} - \frac{1}{2}\mathbf{v}'_{p_{i+1}} = 0, \tag{6}$$

for all vertices in $\mathbf{V}_p$ excluding the begin and end vertices. The averaging constraint *loosens* the positional constraint, allowing edge path vertices to move between their adjacent vertices in the path. Adjusting the ratio of weights between positional and averaging constraints leads to a trade-off between accurately approximating the sketch, and some possibly desired path smoothing.

We have experienced no detrimental effects when applying this procedure on meshes which approximate the underlying smooth surface well, even in areas of high curvature. Also, small changes might be tolerable, as this region will be subsequently edited.

After the geometry adjustment step, the surface is prepared for editing operations in the vicinity of the sketch.

4.2 Sharp features

To create a sharp feature along the edge path, we adjust the Laplacians of path vertices when constructing the $\mathbf{A}$ matrix by prescribing the Laplacian transform for sketch vertices without flexibility to rotate or scale (i.e., as in Eq. 4). Since we discretize the Laplacian using the cotangent weights, we can simply scale the Laplacians of edge path vertices, resulting in a ridge or ravine, depending on the sign. If the Laplacian evaluates to zero, as is the case for flat surfaces, we instead scale the surface normal and prescribe it as the new Laplacian. As described in Section 3, we factor the matrix $\mathbf{A}^T\mathbf{A}$ once we have selected a sketch, and can then quickly evaluate the results of varying scales by dragging the mouse up and down. The creation of a sharp ridge is shown in Fig. 6(d). Alternatively,

we can add some amount to the Laplacians, making the change absolute rather than relative. This works well in regions with high curvature variation along the sketch.

We have found it to be very convenient to create a ridge using our modeling framework, and thereafter treat it as a silhouette from a different camera position and edit it as outlined in Section 3. This technique was applied in the creation of the wavy ridge along the nose of the CAMEL model in Figures 1 and 2(7).

4.3 Smooth features and suggestive contours

Applying the editing metaphor described in the previous section can only create sharp features. To enable smooth features or suggestive contours, we need to influence the Laplacians of more vertices than only those lying on the edge path. Additionally, for suggestive contours, we intend to manipulate curvature in the viewing direction. Thus, we need to rotate the Laplacians w.r.t. an axis which is orthogonal to both viewing and normal vectors. After performing the geometry adjustment of Section 4.1, given the viewing position $\mathbf{c}$, we gather and segment vertices within a user-defined sketch region around the edge path as follows (Fig. 8, top):

- For each path vertex $\mathbf{v}_{p_i}$ with normal $\mathbf{n}_{p_i}$ (the yellow vectors in Fig. 8) we compute the radial plane $\mathbf{r}_i$, which passes through $\mathbf{v}_{p_i}$ with plane normal $\mathbf{n}_{r_i} = (\mathbf{v}_{p_i} - \mathbf{c}) \times \mathbf{n}_{p_i}$ (the blue vectors in Fig. 8). Now we can segment the vertices in the sketch region $\mathbf{V}_s = (\mathbf{v}_{s_1}, \mathbf{v}_{s_2}, \ldots, \mathbf{v}_{s_n})$ such that each sketch region vertex is associated with one such plane (ergo, each vertex in $\mathbf{V}_s$ *belongs* to one edge path vertex).
- Each vertex in $\mathbf{V}_s$ is assigned to the radial plane it is closest to, where the distance of $\mathbf{v}_{s_j}$ to plane $\mathbf{r}_i$ is measured as $d_j = orthodist(\mathbf{r}_i, \mathbf{v}_{s_j}) + dist(\mathbf{v}_{p_i}, \mathbf{v}_{s_j})$. Here, *orthodist* measures orthogonal distance to the plane, and *dist* is the Euclidean distance between $\mathbf{v}_{p_i}$ and $\mathbf{v}_{s_j}$. We take Euclidean distance into account to avoid problems which occur when two different path vertices have similar radial planes, and furthermore to limit the support of the sketch region.

In Fig. 8 (top image), we show one such segmentation, where the edge path vertices are highlighted with red circles and the segmentation is color coded (i.e. all vertices of the same color are associated with the path vertex of that color).

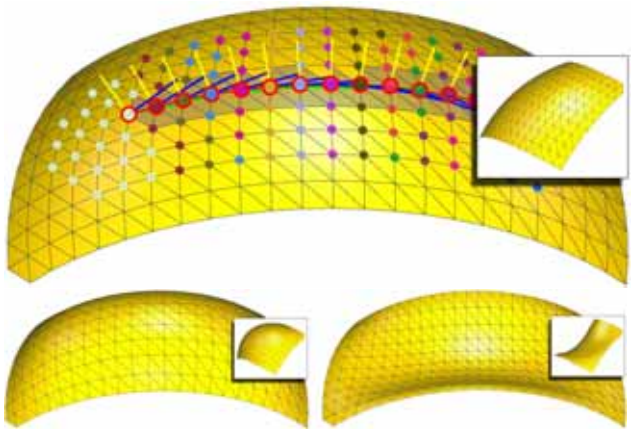

Figure 8: Top: view dependent vertex segmentation and rotation axis assignment. Bottom left: scaling all Laplacians in the sketch region by the same factor produces smooth ridges and ravines. Bottom right: rotating all Laplacians by an angle of $-\pi/2$ w.r.t. the blue rotation axes results in a suggestive contour.

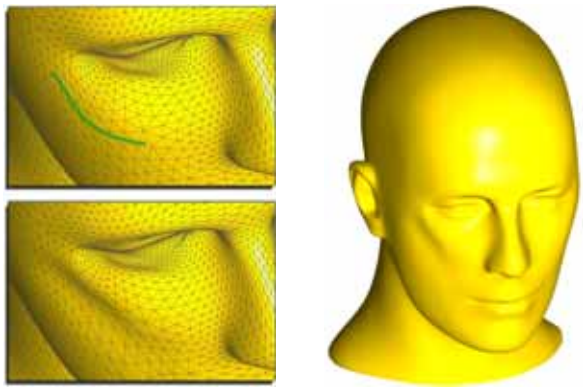

Figure 9: Adding a strong cheekbone to the MANNEQUIN model by sketching a suggestive contour.

Once we have this segmentation, one possible operation is to uniformly scale (or add to) the Laplacians of all sketch region vertices. Complementing the sharp features of Section 4.2, this operation gives us smooth bumps and valleys (Fig. 8, bottom left). By setting the Laplacians to zero we can flatten specific regions of the mesh.

An alternative editing behavior results from rotating all Laplacians w.r.t. their respective rotation axes (given by above segmentation) by a user-defined angle, determined by dragging the mouse left or right. Note that rotation by π is identical to scaling by minus one. For angles in the ranges $[0, \pi)$ and $(\pi, 2\pi]$ we create varying radial curvature inflection points (Fig. 8, bottom right), resulting in suggestive contours [DeCarlo et al. 2003] such as the cheekbone shown in Fig. 9. Note that these inflection points are not necessarily directly under the sketch, since they result from the Laplacian surface reconstruction and the boundary constraints around the ROI.

5 Discussion

Generating plausible and visually pleasing shapes and deformations is far from trivial: while our capability to derive a mental model from everyday shapes around us is well developed, we fail to properly communicate this to a machine. This is why we have to model in a loop, constantly correcting the improper interpretation of our intentions.

The quality of shape editing, therefore, depends on two factors: the time required by the system to update the shape after user commands and how well the shape change reflects our mental model of that process. The update time is a potential bottleneck in our approach, as the necessary matrix factorization and back substitution depend on the number of vertices and not the complexity of the edit operation. For example, ROI sizes of 5.5K/12K/33K vertices require 0.7/2.5/7.0 seconds for factorization and 0.035/0.07/0.25 seconds for back substitution on an Intel P4/2.0 GHz. On the other hand, we believe we have improved the match between the mental model and shape updates, though this is obviously hard to quantify.

From a user's point of view, our system is similar to other sketch-based editing interfaces [Igarashi et al. 1999; Karpenko et al. 2002; Draper and Egbert 2003; Kho and Garland 2005], while it differs algorithmically: the above methods are based on space warps and variational implicits, whereas our representation is aimed at surface detail preservation. Our method inherits the simplicity of the user interface, and enables the creation of interesting and useful surface edits, both for inexperienced users and modeling professionals.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by

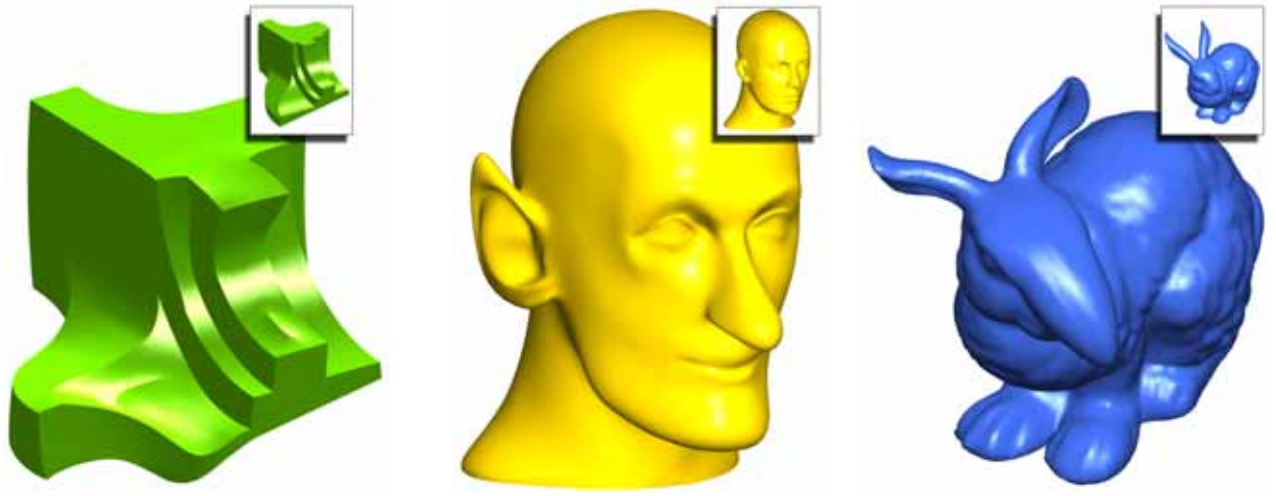

Figure 10: Some results: a deformed FANDISK with a few more sharp features, a rather surprised MANNEQUIN with more than just an extra contour around the eye, and droopy-eared, big-nose BUNNY with large feet.

grants from the European Network of Excellence AIM@SHAPE (FP6 IST NoE 506766), the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities) and by the Israeli Ministry of Science.

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

BIERMANN, H., LEVIN, A., AND ZORIN, D. 2000. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of SIGGRAPH 2000*, 113–120.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOURGUIGNON, D., CHAINE, R., CANI, M.-P., AND DRETTAKIS, G. 2004. Relief: A modeling by drawing tool. In *First Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 151–160.

DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3, 848–855.

DRAPER, G., AND EGBERT, P. 2003. A gestural interface to free-form deformation. In *Proceedings of Graphics Interface 2003*, 113–120.

FLEISCH, T., RECHEL, F., SANTOS, P., AND STORK, A. 2004. Constraint stroke-based oversketching for 3D curves. In *First Eurographics Workshop on Sketch-Based Interfaces and Modeling*.

GOOCH, B., AND GOOCH, A. 2001. *Non-Photorealistic Rendering*. A.K. Peters.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proceedings of SIGGRAPH 2000*, 517–526.

HERTZMANN, A. 1999. Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. In *Non-Photorealistic Rendering. SIGGRAPH 99 Course Notes.*

HOFFMAN, D. D., AND SINGH, M. 1997. Salience of visual parts. In *Cognition*, vol. 63(1), 29–78.

IGARASHI, T., AND HUGHES, J. F. 2003. Smooth meshes for sketch-based freeform modeling. In *2003 ACM Symposium on Interactive 3D Graphics*, 139–142.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH 99*, 409–416.

KARPENKO, O., HUGHES, J. F., AND RASKAR, R. 2002. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3, 585–594.

KHO, Y., AND GARLAND, M. 2005. Sketching mesh deformations. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, 147–154.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH 98*, 105–114.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., AND LEVIN, D. 2004. Differential coordinates for interactive mesh editing. In *International Conference on Shape Modeling and Applications*, 181–190.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III*, pages 35–57.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, vol. 20, 151–160.

SINGH, K., AND FIUME, E. L. 1998. Wires: A geometric deformation technique. In *Proceedings of SIGGRAPH 98*, 405–414.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, 179–188.

SUMNER, R., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3, 399–405.

TOLEDO, S. 2003. TAUCS*: A Library of Sparse Linear Solvers*. Tel Aviv University.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. Sketch: An interface for sketching 3D scenes. In *Proceedings of SIGGRAPH 96*, 163–170.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proceedings of SIGGRAPH 97*, 259–268.