Image and 3D-Object Editing with Precisely Specified Editing Regions

Gerhard Bendels *
Computer Graphics
University of Bonn, Germany

Reinhard Klein †
Computer Graphics
University of Bonn, Germany

Andreas Schilling ‡
Media Informatics
Hochschule der Medien, Stuttgart, Germany

Abstract

In this paper we present a new approach to the editing of polygons, images and triangulated surface meshes featuring the exact specification of the area that is being edited, the area that remains unchanged and an area that is transformed rigidly. This allows intuitive editing of arbitrary triangle meshes as well as 2D-images. Furthermore, the shape of the deformed area can be modified interactively by manipulating a simple parametric curve. The key contributions of this paper are that we include not only translations and rotations in our editing paradigm but the whole spectrum of affine transformations. We include a refinement approach to allow for sharp feature edits even in sparsely sampled areas. In spite of these features the algorithm is very fast and simple and can easily be generalized to the case of tetrahedrized volumes.

1 Introduction and previous work

The growing availability of low-cost scanning devices and the advances in reconstruction techniques deliver a huge collection of 3D-Models on-hand in various data bases, e.g. the world wide web. At the same time, the focus of much research shifts from design methodologies fulfilling industry imposed requirements like exact and provable geometric properties (as in [18, 12]) to the development of simple and intuitive design tools, that are required in numerous computer graphics applications like the film industry, computer games, etc.

A very prominent and well-known example of modelling paradigms are the implicit deformations where the objects are embedded in a space. The warping of this space causes a corresponding deformation of the object. According to the most popular such approach called Free Form Deformations (FFD) [4, 40], the surrounding space is defined as a multidimensional spline. This technique has been further improved and generalized [15], moreover, it was adapted to generate animations by e.g. [15, 17]. The main advantage of these methods is that they enable a deformation that is independent of the complexity of the object being manipulated. However, as pointed out in [19], the placement and control of the lattice defining the deformation is non-trivial. The improvements of [19, 20] remedied these problems for the case of a single edit.

The classical NURBS modelling approach [18] also falls into the category of implicit modelling. It reflects the idea of imposing a control mesh directly on the surface of the object, the desired modelling effect is then achieved by modifying the control points of this mesh. However, since there are few degrees of freedom in determining the scale of an edit and the topology of the control mesh, working with NURBS requires much expertise and is time intensive.

The algorithmic generalization of piecewise polynomial representations are the subdivision surfaces [13, 12], where a smooth surface is generated by iterative refinement of a control mesh according to certain subdivision rules. Since subdivision surfaces essentially comprise a representation of an object on different levels of scale, they may be utilized as a basis for Multiresolution Mesh Editing (MME) [45]. The idea of multiresolution editing is to use different levels of detail of the object to perform edits on different scales: Detail edits are performed

*bendels@cs.uni-bonn.de
†rk@cs.uni-bonn.de
‡schilling@uni-tuebingen.de

on finer meshes and large scale edits on coarser meshes representing the same object. Saving the finer meshes as details with respect to the coarser meshes provides for detail preservation during large scale edits. Unfortunately, since the one-ring of the edited vertex defines the region of influence (ROI) of an edit, the subject of the edit and the ROI are completely connectivity-defined and cannot be chosen arbitrarily. Kobbelt et al. [29, 30] provided a solution to this problem by utilizing a hierarchy of scales in terms of smoothness. The levels of detail are generated by a hierarchical mesh smoothing scheme. During the editing process, the area of influence of the editing operation can be specified e.g. by defining two boundary curves that are drawn on the triangle mesh, a feature that is indispensable for intuitive editing. The scale of the edit is implicitly defined by the specification of the region borders. Lacking appropriate basis functions, Kobbelt et al. modified the deformable area by an energy minimization technique in the spirit of Welch and Witkin [42]. The shape of the resulting surface mesh is predetermined by this technique.

Another solution was presented by Lee [32]. He allows a simple connected area of the mesh to be defined as editing area. This area is embedded onto a rectangle of the 2D-plane using harmonic maps [16]. In this way he reduces the 3D editing to B-Spline surface manipulation in 2D. Unfortunately, in addition to the problems with calculating the harmonic maps, the deformation of an arbitrarily shaped area to a rectangle in many cases misleads intuition during editing.

Despite the shortcomings – effectively digitizing and interactive display is still a challenge – implicit modelling was subject to extensive research in recent years. Distance surfaces like *blobs* [6], *metaballs* [9], *soft objects* [8], and *convolution surfaces* [7] are popular in Computer Animation since the geometric ”skeleton”, with respect to which they are defined, can be used as an internal structure to control the animation [10] and even for LOD-representations [11, 2]. Several methods have been presented to tackle the so-called ”unwanted blending problem”, e.g. [3]. Furthermore, the ubiquitous availability of inside-outside information allows for efficient collision detection and response [38] and accounts for the development of intuitive haptic editing interfaces, e.g. [21, 22].

While the methods mentioned so far mainly apply to modifying existing shapes, SKETCH [44], SKIN [35] and TEDDY [23], which were later extended in [25], are among those methods generating models from scratch. Modifying shapes in these approaches is realized using a method called oversketching, i.e. drawing parts of the silhouette of the shape anew.

Although triangle meshes are still the key 3D object representation in Computer Graphics due to their extensive hardware support, the manipulation of unstructured point data gained more and more research attention in recent years. Zwicker et al. [46] generalize standard 2D image editing techniques to 3D, reconstructing well-known pixel editing tools. Of particular relevance for this paper is the work of Pauly et al. presented most recently in [39]. Here, multiresolution results are transferred to the case of point-sampled geometry. In extension of a preliminary idea presented at [27], shapes are modified by defining a so-called *zero-region* and a *one-region*. The one-region undergoes the full user-defined translation or rotation rigidly, whereas the zero-region remains fixed and a pre-defined blending function is used to create a smooth transition between the two regions. Similar in concept, Llamas et al. [33] let users define transformations for single handle vertices on the surface. These transformations are interpreted as a time-dependent screw motion. The handle vertices undergo the full screw motion, while vertices in the neighborhood move along an identic path but not to the full extent. The required time parameter for every vertex on the surface is also computed using a pre-defined blending function.

In contrast to these approaches we solve the problem of smoothly propagating the handle transformation to the neighborhood by applying recent advances [1] in the linear combination of transformations. This allows for the whole spectrum of affine transformation to be smoothly applied in the editing region in a consistent and analytic way. We utilize geodesic distances on the mesh (unavailable for point clouds) to compute the blending function values.

In [39] a twofold refinement strategy is proposed, in order to be able to reflect sharp features generated by CSG operations, and also to tackle insufficient surface sampling caused by extensive surface stretch. These refinement methodologies rely on a sufficiently high sampling rate before the edit starts,

so that point positions can be interpolated linearly, leaving these strategies inapplicable in the case of much more sparsely sampled triangle meshes. Therefore, we use a refinement strategy that allows fine detail edits even in sparsely triangulated areas, following the idea from [5].

Moreover, we show by several examples that our editing scheme is not only applicable to 3D-Meshes but also constitutes a powerful ingredient of a 2D image editing toolbox.

In section 2 we briefly present the general ideas and definitions for the case of triangle meshes. The key to the new method is the propagation of the transformation to the neighborhood of the handle. This is subject of section 3. The definition and computation of appropriate distance fields in 3D is discussed in section 4.

2 The basic idea

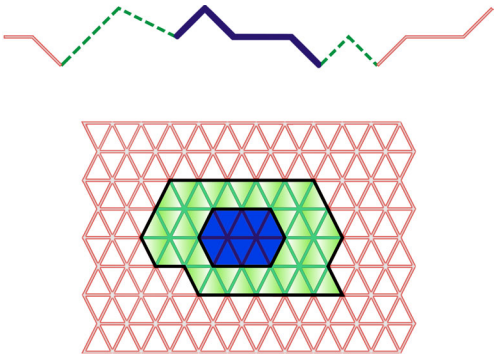

Figure 1: Partition of a polyline (top) and a triangle mesh (bottom) into *handle* (dark blue), *deformable* (dashed green), and *fixed* area.

Similar to the concept presented in [27] and [39], the first step of the algorithm is to split the object to be edited into disjoint parts, defined by its borders that are drawn onto the triangle mesh using a special 3D-cursor: The *fixed area* and the *handle*, see Fig. 1. The former consists of all vertices in the triangle mesh whose position remains unchanged during editing, whereas the latter consists of all vertices that undergo the same transformation, thereby defining an area that is transformed rigidly. The *deformed area* consists of the remaining vertices between the stiff area and the handle.[1]

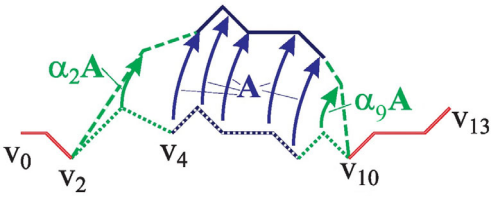

Figure 2: Transformation of a simple polygonal curve. Vertices v_4 to v_8 form the handle and undergo the full transformation A, whereas vertices v_0 to v_2 and v_{10} to v_{13} remain fixed.

The basic idea of the editing algorithm can best be explained in the case of a onedimensional polygonal curve, see Fig. 2: If the handle is moved (by applying the transformation $\mathbf{A}$), we have to decide what happens to the vertices in the deformed area. We solve the problem by applying the scaled transformation $\alpha_i \odot \mathbf{A}$ to the vertices v_i of the deformed area (see section 3), with the property that for $\alpha = 0$ the resulting transformation is the identity and for $\alpha = 1$ the transformation is $\mathbf{A}$ itself.

In our setting, $\alpha_i \in [0, 1]$ depends on the distance $d_{i,\text{fixed}}$ of the the regarded vertex v_i to the fixed area and the distance $d_{i,\text{handle}}$ to the handle. Generally, if the vertex v_i is close to the fixed area then α_i is close to 0, and close to 1 if v_i is close to the handle.

Of course, this basic editing idea is not restricted to polygonal curves but it can also be applied to images, triangle meshes, and control nets of higher order surfaces (this way, groups of control points can be transformed and deformed in an intuitive manner) as well, given that appropriate distances from the respective primitives to the corresponding borders on the object can be calculated (see section 4).

3 Transformations

A transformation matrix $\mathbf{A}$ can intuitively be defined for the handle vertices using a 3D-input device or a 2D-mouse by first specifying the origin of the coordinate system of the transformation. Then a point on the handle is picked and can be dragged, see Figure 8 for an example.

[1] We expect the handle to be completely contained in the complement of the fixed area.

This way, the transformation is defined explicitly for the handle vertices only. For the vertices in the deformed region, the transformation has to be determined. In order to achieve smooth editing operations, we want the vertices in the deformed region to be transformed by a scalar power $\alpha_i \odot \mathbf{A}$ of the transformation $\mathbf{A}$ defined for the handle.

As stated above, for $\alpha = 0$ we want the resulting transformation to be the identity, for $\alpha = 1$ the transformation should be $\mathbf{A}$, and in the particular case $\alpha = \frac{1}{2}$, i.e. 'half' of $\mathbf{A}$, we want the resulting transformation to have the property that, being applied twice, the result is the original transformation.

Defining the scalar power of a transformation $\mathbf{A}$ is straightforward in the case of translations, rotations, and scaling. (If the latter is defined by a scale-matrix with diagonal entries d_1, d_2, d_3, then $diag(d_1^\alpha, d_2^\alpha, d_3^\alpha)$ can be used.)

In contrast, interpolating transformations in the case of an arbitrary transformation is non-trivial. In [33], this problem is tackled by computing a time-dependent minimal screw motion. Applying $\frac{1}{2} \odot \mathbf{A}$ is then interpreted as following the exact same screw motion *half way*.

Instead, we follow the approach of Alexa [1], who used the exponential and logarithmic function to define[2]

$$\alpha \odot \mathbf{A} = e^{\alpha \log \mathbf{A}}.$$

For a single editing operation the logarithm $\log \mathbf{A}$ of the transformation matrix $\mathbf{A}$ has to be computed only once, but the exponentiation $e^{f(d_{out},d_{in}) \log \mathbf{A}}$ has to be performed for every vertex in the deformed area. As reported by Alexa [1] the exponentiation is about 10 times faster than the computation of the logarithm and takes about $3 \cdot 10^{-6}$ sec on a 1GHz Athlon. Therefore, interactive frame-rates are achievable even for huge numbers of vertices in the deformed area.

4 The distance field

As in [27] we choose α to be a function $f : [0, 1] \rightarrow [0, 1]$ of the ratio

$$\frac{d_{i,\text{fixed}}}{d_{i,\text{fixed}} + d_{i,\text{handle}}}.$$

[2]The logarithm is well-defined for transformation matrices, as long as the transformation contains no reflection. See [1] for details.

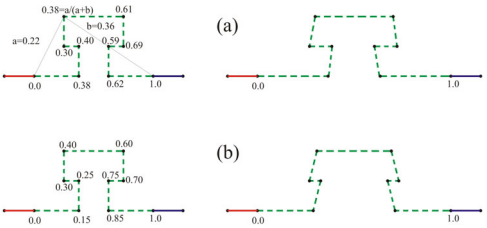

Figure 3: (a) Computation of α using Euclidean distances (a), and arc length parametrization (b). Note that in contrast to (b), α does not grow monotonically in (a) when going from vertex to vertex towards the handle, leading to a different propagation of the transformation defined for the handle (translation in x-direction).

This way, the *distance fields* defined by the border of the handle and the border of the fixed area deliver us a natural parameterization of the deformed area. The function f defines the resulting shape of the deformed area and is therefore called *shape function*.

Since the choice of a specific distance field has a strong impact on the editing operations we consider two different distance fields: The Euclidean distance field defined over the Euclidean space and the geodesic distance field defined on a polygon or on the triangle mesh.

Using the efficient and ubiquitous Euclidean distance for editing meshes in 3D results in an editing behavior that resembles in some way the classical space deformation techniques [4, 40, 15, 19, 34], with the important advantage that, with our approach, the edited area can exactly be defined on the triangle mesh by defining the borders of fixed area and handle. Furthermore, the problems of classical space deformation techniques with a "continuous" connection between modified and fixed area can easily be avoided by choosing appropriate shape functions.

Using geodesic distances on the other hand allows taking into account object surface properties for the influence computation and leads to an editing behavior that resembles the well known spline editing.

Although the distance field computation must be performed only once at the beginning of each editing step, having fast algorithms to compute the Euclidean as well as the geodesic distance field is crucial for interactive editing.

4.1 Euclidean distances

For each vertex v, the Euclidean distances to the handle h and the fixed area f are given by

$$d_E(v,h) = \min_i(d(v,(v_iv_{i+1})) \quad (1)$$

$$d_E(v,f) = \min_j(d(v,(v_jv_{j+1})) \quad (2)$$

where (v_iv_{i+1}) is the line segment between vertex v_i and v_{i+1} on the border of the handle and (v_jv_{j+1}) is the line segment between vertex v_j and v_{j+1} on the border of the fixed area.

Although the computation is not complicated, it is expensive for larger border polygons as the complexity of a simple algorithm grows linearly with the size of the border polygons. To accelerate the computation, a spatial search data structure supporting nearest simplex queries is necessary. This data structure must be dynamic as in each editing step new border polygons are defined. We use a regular grid for the spatial search because it optimally performs in static and dynamic environments [43] and is easy to implement. In each grid cell we store a list of the border edges partially or completely contained in the cell. The edges are inserted into the grid by using a simple incremental algorithm which traces the penetrated cells along the edge. In the beginning of the editing process we use a grid with uniform edge length of twice the average edge length of the model, such that each edge is in average contained in one or two cells allowing for fast insertion and removal. While editing continues we keep track of the decreasing average edge length and refine the grid if necessary.

In the case of images we exploit graphics hardware [28, 24] to compute the Euclidean distance field, which in this case coincides with the geodesic one. The basic idea in this context is to define objects, whose z-values are proportional to the distance we want to compute and render them into the z-buffer. For a single vertex the needed object is a cone, for a line two half-planes are used. As the cones can only be approximated by polygons special techniques are needed to keep the results consistent. A simple solution to this problem is discussed in detail in [28].

4.2 Geodesic distances

For each vertex in the deformable area, the two geodesic distances are defined by the length of the shortest path to the respective border.[3] The algorithms to compute geodesic distances elaborated so far are usually formulated for convex polytopes, which is sufficient for a number of applications. The solution by Shahir and Schorr [41] yields an algorithm with $O(n^3 \log n)$ complexity; an important additional result here is the proof that every shortest path on a polyhedron unfolds to a straight line on a plane. Mitchell et al. [36] showed an improved $O(n^2 \log n)$ running time algorithm working on non-convex polyhedra as well. The best algorithm known to the authors for the case of non-convex polyhedra was presented by Chen and Han [14] with $O(n^2)$ time complexity.

Figure 4: A gap in the hippo mesh on the right leads to unexpected distance values and therefore to non-intuitive editing results, whereas the left side hippo's mesh is 2-manifold and the geodesic distance field behaves as supposed.

For applications preferring speed over accuracy in the calculation of the geodesic distances, approximative algorithms are used. For the general case of non-convex objects such algorithms were presented in [31]. In this work we use an improved version of the algorithm of Kimmel and Sethian [26] described in [37].

5 The shape function

The shape function $f : [0,1] \longrightarrow [0,1]$ can be chosen arbitrarily. Generally, though, it is desirable to postulate continuity properties of the shape function.

Although the properties of continuity and differentiability are not directly applicable to polygons it is obvious that if f is continuous, $f(0) = 0$ and $f(1) = 1$, the result of the deformation will still be "continuous" (in the limit of infinite subdivision of the polygon, we could drop the quotes). An analogue argument shows that if in addition to the above conditions, f is differentiable, $f'(0) = 0$

[3] Naturally, this definition is only useful if the mesh to be edited is 2-manifold, see figure 4.

and $f'(1) = 0$, we will get a "differentiable" deformation result. The same is true for higher order derivatives.

Compared to classical CAD approaches, e.g. modelling with B-Spline-tensor product surfaces, our approach has two main advantages:

1. the shape function can be chosen by the user and is not implicitly given by the representation of the surface.
2. the area of influence of the shape function can be freely defined and precisely specified by the user. In the case of B-Spline-tensor product surfaces this is not possible and in most cases the user does not even know a priori the area of influence.

As in [5] it is thus possible with our method to perform editing with a two-step approach: In the first step, the user moves the handle to the desired location and in the second step he determines the final shape of the deformable area by fine-tuning the deformation function.

The one-dimensional shape function determines the shape of the modified area everywhere around the handle in the same way based on the ratio of the two distance values. Sometimes it might be desirable to modify the deformed area around the handle in different ways. To this end, a two-dimensional shape function would be necessary. An easy solution would be to manually draw lines that connect the two borders of the deformable area on which the deformation functions can be specified. Between these lines the applied deformation function is again interpolated according to the distances to these lines.

6 Refinement

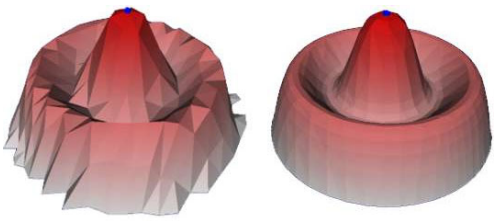

Figure 5: For some edits, the triangles might be too large to represent fine details as defined by the shape function (left). Result after subdivision (right).

If the deformation becomes too large, it can happen that even fine meshes finally contain very large triangles. In this case, even with a smooth shape function, no smooth resulting mesh is achievable. Therefore we recursively apply Rivara Bisection where required, i.e. we subdivide the large triangles into four triangles by inserting the midpoints of its three edges into the mesh, see figure 5. To compute the position of the transformed new vertices we must know their distances to the borders of the fixed area and the handle. A simple linear interpolation of the original distance values at the vertices of a subdivided triangle is generally not sufficient. Instead, in [5] we propose approximating the geodesic distance for every new vertex v on the edge (v_0, v_1) from the distance values δ_0 and δ_1 of the two adjacent vertices. To this end, a virtual origin for the distance calculation is computed on the intersection of the two spheres with radii δ_0 and δ_1 and centers v_0 and v_1, resp. The distance to this virtual origin is used as an approximation of the geodesic distance value for v.

7 Interactive editing examples

The first example (Fig. 6 (a) and (b)) shows the translation and the rotation ((c) and (d)) of a stiff handle in an image. The parts of the image that are to be modified can be specified precisely. Note that all deformations can be performed interactively by dragging with the mouse even on low-powered PCs.

Figures 7 and 8 show images that have been captured during interactive editing of 3D triangle meshes. See the captions below the figures for explanations.

8 Conclusions and future work

We have presented a simple and efficient high level editing algorithm for polygons, images and arbitrary triangle meshes in 3D. The two main features of our algorithm are that we can specify an exact area of influence for the editing process and that the shape functions can be chosen arbitrarily. The transformation specified for the handle region is appropriately propagated into the deformable area of the object using (depending on the actual editing circumstances) geodesic or Euclidean distances. The scalar power of a transformation required for

this propagation is applied according to recent advances in the computation of linear combinations of transformations. This allows for arbitrary transformations to be smoothly interpolated in the deformed regions and is therefore superior to comparable approaches. Moreover, we include a refinement strategy into our editing environment to be able to approximately model sharp features. This could be further improved by taking into account the shape function features as was done in [5]. We therefore consider our algorithm a useful tool in an editor's toolbox.

In our current implementation the border lines are restricted to the edges in the mesh. A straightforward extension of our algorithm is therefore to allow arbitrary lines on the object surface. This would lead to an even more flexible editing approach. Moreover, we are currently working on an extension of the one-dimensional shape function to two dimensions. The next step is an implementation of the algorithm on tetrahedral meshes. The extension of the distance field calculations to this case is also straightforward.

Acknowledgements

We would like to thank Holger Müller who implemented the Curve- and Image-Editing algorithm and Pavel Borodin and Michael Guthe who worked on the 3D Mesh Editing algorithm.

References

[1] M. Alexa. Linear combination of transformation. In *Computer Graphics (SIGGRAPH 2002 Proceedings)*, pages 380–387, July 2002.

[2] A. Angelidis and M.-P. Cani. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Solid Modelling and Applications*. ACM, June 2002. Saarbrucken, Germany.

[3] A. Angelidis, P. Jepp, and M.-P. Cani. Implicit modeling with skeleton curves: Controlled blending in contact situations. In *Shape Modeling International*. ACM, IEEE Computer Society Press, 2002. Banff, Alberta, Cananda.

[4] A. H. Barr. Global and local deformations of solid primitives. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, 1984.

[5] G. H. Bendels and R. Klein. Mesh forging: Editing of 3d-meshes using implicitly defined occluders. In *Proceedings of the Eurographics Symposium on Geometry Processing 2003*, June 2003.

[6] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982.

[7] J. Bloomenthal and K. Shoemake. Convolution surfaces. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 251–256. ACM Press, 1991.

[8] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):109–116, 1990.

[9] B.Wyvill, C.McPheeters, and G.Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[10] M.-P. Cani. Implicit representations in computer animation : a compared study. In *Proceedings of Implicit Surface '99*, Sep 1999. Invited paper.

[11] M.-P. Cani and S. Hornus. Subdivision curve primitives: a new solution for interactive implicit modeling. In *Shape Modelling International*, Italy, May 2001.

[12] E. Catmull and J. H. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–360, November 1978.

[13] G. Chaikin. Short note: An algorithm for high-speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[14] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Symposium on Computational Geometry*, pages 360–369, 1990.

[15] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196. ACM Press, 1990.

[16] M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 173–182, Aug. 1995.

[17] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, /1997.

[18] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press Inc., 1993.

[19] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.

[20] S.-M. Hu, H. Zhang, C.-L. Tai, and J.-G. Sun. Direct manipulation of ffd: efficient explicit solutions and decomposible multiple point constraints. *The Visual Computer*, 17, 2001.

[21] J. Hua and H. Qin. Haptic sculpting of volumetric implicit functions. In *Proceedings of 9th Pacific on Computer Graphics and Applications*, pages 254–264, 2001.

[22] J. Hua and H. Qin. Haptics-based volumetric modeling using dynamic spline-based implicit functions. In *Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, pages 55–64. IEEE Press, 2002.

[23] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM Press/Addison-Wesley Publishing Co., 1999.

[24] K. H. III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 277–286, Aug. 1999.

[25] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3), September 2002.

[26] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of National Academy of Sciences*, 95(15):8431–8435, 1998.

[27] R. Klein. 3d mesh-editing. In *Dagstuhl Seminar: Image Synthesis and Interactive 3D Graphics*, volume 25. Michael Cohen, Heinrich Mueller, Claude Puech, Hans-Peter Seidel, June 2000 – http://cg.cs.uni-bonn.de/docs/publications/2000/dagstuhl-presentation.pdf.

[28] R. Klein and A. Schilling. Fast distance field interpolation for reconstruction of surfaces from contours. In *Eurographics '99, Short Papers & Demos proceedings*, 1999. Conference held in Milano, Italy,.

[29] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *Computer Graphics*, 32(Annual Conference Series):105–114, 1998.

[30] L. P. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3), August 2000.

[31] M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *6th Annual Video Review of Computational Geometry, Proc. 13th ACM Symp. Computational Geometry*, pages 485–486. ACM Press, 4–6 1997.

[32] S. Lee. Interactive multiresolution editing of arbitrary meshes. In P. Brunet and R. Scopigno, editors, *Proc. of Eurographics '99*, pages C–73–C82, 1999.

[33] I. Llamas, B. Kim, J. Gargus, J. Rossignac, and C. D. Shaw. Twister: A space-warp operator for the two-handed editing of 3d shapes. *ACM Transactions on Graphics*, 22(3):663–668, July 2003.

[34] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 181–188, Aug. 1996.

[35] L. Markosian, J. M. Cohen, T. Crulli, and J. Hughes. Skin: a constructive approach to modeling free-form shapes. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 393–400. ACM Press/Addison-Wesley Publishing Co., 1999.

[36] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.

[37] M. Novotni and R. Klein. Computing geodesic distances on triangular meshes. In *The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 341–347, 2002.

[38] A. Opalach and M. Cani-Gascuel. Local deformations for animation of implicit surfaces. In W. Straßer, editor, *13th Spring Conference on Computer Graphics*, pages 85–92, 1997.

[39] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics*, 22(3):641–650, July 2003.

[40] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160. ACM Press, 1986.

[41] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM Journal on Computing*, 15(1):193–215, 1986.

[42] W. Welch and A. Witkin. Variational surface modeling. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, volume 26, pages 157–166, July 1992.

[43] G. Zachmann. Optimizing the collision detection pipeline. In *The First International Game Technology Conference GTEC*, Jan. 2001.

[44] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: an interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 163–170. ACM Press, 1996.

[45] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics*, 31(Annual Conference Series):259–268, 1997.

[46] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 322–329. ACM Press, 2002.

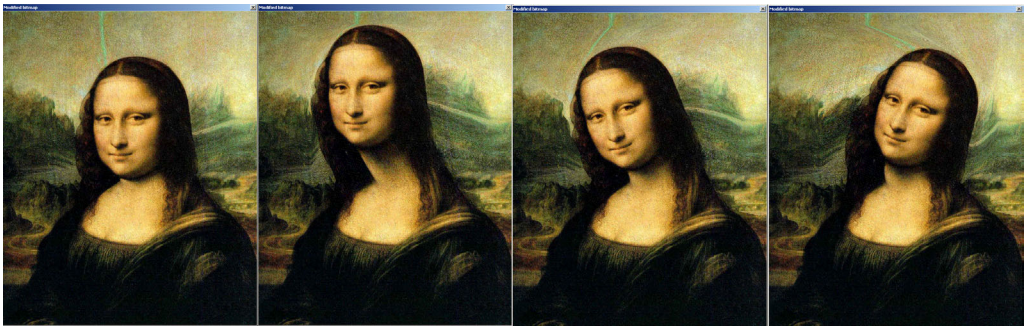

Figure 6: Image editing: Translation ((a) and (b)) and rotation ((c) and (d)) of the head of da Vinci's Mona Lisa.

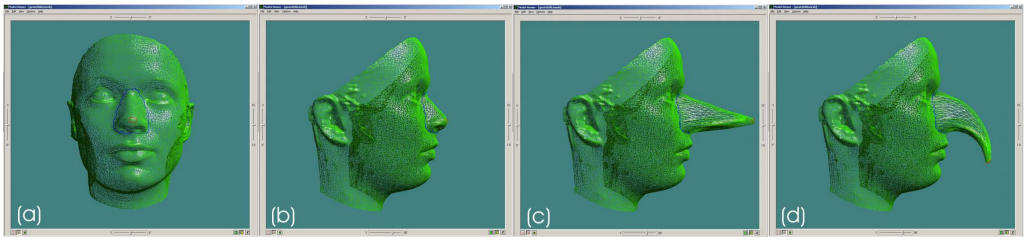

Figure 7: Mesh editing example. (a) Definition of fixed area and handle. Note that the border of the handle is chosen casually. In the example the handle area contains only a few triangles. (b) Side view of the model before deformation. (c) The tip of the nose is dragged; in the silhouette it can be seen that the connections between deformed area and fixed area or handle, respectively, remain "continuous". (d) Instead of a translation a rotation is applied. Note that these deformations are interactively performed on a simple 500 MHz Laptop.

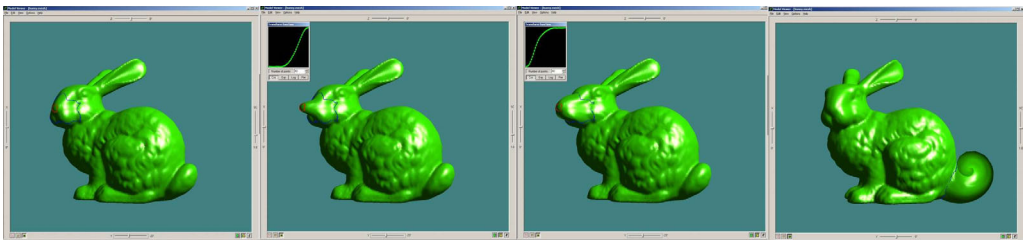

Figure 8: Using the shape function. First of all, the handle (in this case the nose) is dragged. Then a shape function is interactively modified as shown in the middle figures. In the middle left example larger displacements occur only close to the handle, resulting in a pointed nose. In the middle right example the shape function raises already at small values and therefore the deformation starts close to the fixed area. The picture to the right shows a fancy editing of the tail of the Stanford bunny using a rotation.