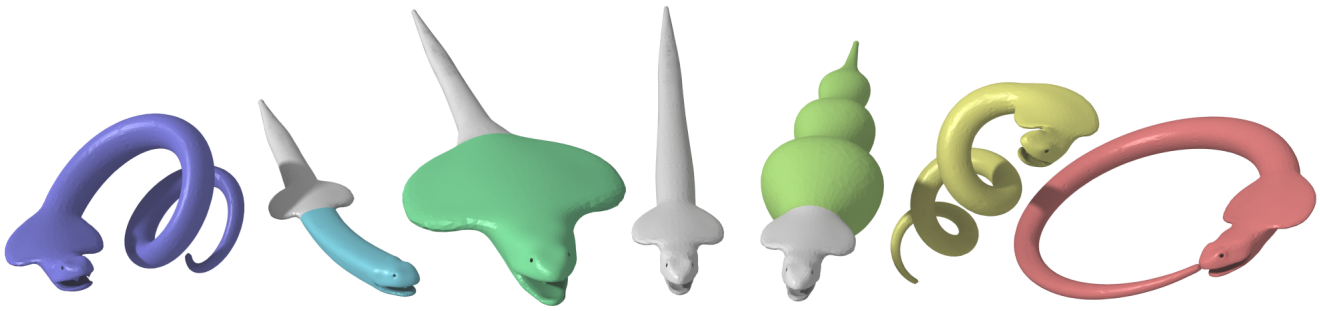
Noname manuscript No.
(will be inserted by the editor)

Differential-Based Geometry and Texture Editing with Brushes

Niklas Krauth · Matthias Nieser · Konrad Polthier



Received: date / Accepted: date

Abstract We present an interactive modeling framework for 3D shapes and for texture maps. The technique combines a differential-based deformation method with the idea of *geometry brushes* that allow to interactively apply modifications by *painting* on the geometry. Whereas most other deformation techniques demand the designer to define and move hard constrained regions on the surface, the proposed modeling process is similar to sculpting.

Geometry brushes allow the user to locally manipulate the metric, enlarge, shrink or rotate parts of the surface and to generate bumps. In a similar way it is possible to modify texture maps, or more generally, arbitrary tensor maps on surfaces. The local modifications of the surface are integrated to a globally consistent deformation and visualized in real-time.

While the geometry brushes are intended for local editing, the underlying technique can also be applied globally. We show how differentials may be modified for creating specific effects, like cartoonization of shapes or adjusting texture images.

Keywords geometry brush · deformation · interactive modeling · texture editing

Niklas Krauth · Matthias Nieser · Konrad Polthier
Freie Universität Berlin,
Arnimallee 6, 14195 Berlin, Germany
E-mail: niklas.krauth@fu-berlin.de, matthias.nieser@fu-berlin.de, konrad.polthier@fu-berlin.de

1 Introduction

Shape deformation is an important field of geometric modeling. Many recent works have focused on surface-based techniques using differential representations and the advantages of this approach are well known. The majority of these works deforms a surface by specifying some hard constraints on the position of several vertices and finding a plausible deformation which respects these constraints. However, for many specific target-shapes it can be tedious to generate the deformation using constraints only.

The most related deformation methods use a handle attached to a subset of vertices which can be manipulated. Usually, these handles can be translated, scaled and rotated. The modifications applied to the handle are then propagated to the rest of the surface. For any small local change the user must first define an appropriate handle before he can perform the manipulation itself. It is often not intuitive how to define the handle for a desired manipulation.

We present a deformation framework based on direct control of the differential Df of a deformation $f : M \to M^*$ between two surfaces. Instead of specifying constraints the designer manipulates the shape locally using geometry brushes. Like a brush in an image processing software, the user can interactively design by painting on the surface. Each brush applies a semi-local transformation to the picked parts, i.e. scale, rotation or any user-defined transformation. By semi-local, we mean that the deformation itself may be global, but some distance away from the selected region, the change

is only marginal. This allows for intuitive editing of a given surface with simple user-interaction.

The framework is based on a method for integrating differentials on a simplicial surface. After altering the differential, either locally or globally, the method generates a feasible global deformation which respects these modifications. It is fast, robust and easy to implement, and allows for deformations in real-time.

In the same way in which the surface itself is deformed, the framework can manipulate any given function defined on the surface, e.g. the UV-coordinates of a texture map. Usually, textures are modeled by creating the texture map once and then by transforming the texture image in $\mathbb{R}^2$ in order to match a given shape. In contrast our framework allows to deform the texture map itself in an intuitive way, e.g. by locally moving, scaling or rotating the texture map while maintaining global consistency.

While the proposed geometry brushes provide a powerful set of transformations to modify a geometry, the technique itself allows for any user-defined transformation of the differential. We show an example of globally modified differentials used to cartoonify the given shape. The surface features are hereby enlarged by scaling the metric dependent on the local curvature, similar in spirit to the curvature clamping and feature enhancement of [5].

1.1 Previous Work

Deformation. The variety of related approaches to surface deformation is quite large. Therefore, we review only the approaches most relevant to ours, namely those based on differential surface representations. For a more comprehensive insight the reader is kindly referred to the excellent overviews by Botsch and Sorkine and by Xu and Zhou [1, 29].

There are several methods which use differential quantities to represent the surface. A deformation can be computed by modifying these quantities and reconstructing the surface. The quantities applied are gradients of the coordinate functions [30], the Laplacian [11, 25], metric plus curvature [5, 4] or a representation with differential forms [16].

All of the methods work by constraining some vertex positions and using the differential representation to compute a nice deformation satisfying the constraints. In contrast, our approach aims at deforming a surface by directly manipulating the differential locally using the proposed geometry brushes. Nevertheless, hard constraints are also incorporated in our framework.

Another technique has been carried over to surface based methods from the realm of space deformations (e.g. [18],[8]). Here, the user modifies only a simple, coarse control-structure and the modifications are in some way propagated to the shape, usually by translating them into constraints. In [31] the user may additionally edit the curvature in precomputed surface regions.

Whereas these metaphors are very well suited to apply large scale changes to a model they are by their nature unable to give the user control over surface details. For example, altering the face details of the Caesar model as depicted in Fig. 7 will be close to impossible with these kinds of methods.

Geometry Brushes. Closer in spirit to the proposed brushes are so called sketching interfaces. Here, the user freely sketches the desired contours of a shape, which is then modified to match those contours [32]. Similarly, he may edit the geometry by drawing a source- and a desired target-line on the surface itself or by prescribing the curvature along a line on the surface [19].

An example of modeling with brushes is the software ZBrush [21]. Here, the user interactively modifies a given geometry by drawing on it using a huge selection of brushes. This is specifically well suited for adding and editing high level detail. The idea of modeling brushes is also presented by Lawrence and Funkhouser [14]. The user paints a deformation on the surface. The painted pattern is then transformed into velocity vectors in normal direction and the deformation is produced by a physical simulation.

The work of von Funck et al. [6] presents a deformation which is obtained by the path line integration of a time-dependent, divergence-free vector field. While their technique supports many more modeling metaphors, it does not allow for scaling by its nature.

Takayama et al. [26] use modeling brushes in order to transfer geometric details from one geometry to another utilizing a differential surface representation.

While all of the above approaches apply brushes to interact with the geometry, the brushes are employed differently.

Maybe, the most similar approach to ours is the work of Crane et al. [2]. They use a brush for painting curvature on a surface and then compute a conformal deformation using the target curvature. By prescribing the curvature the approach controls second derivatives and one can easily add bumps or scale parts of the surface. Our approach gives the user the control over the first derivative which enables different modeling brushes like rotation or anisotropic scaling.

Texture Editing. There is extensive work done for 2D image editing [7, 23, 17], just to name a few. Most often, the deformation is defined by constraining the boundary or special points in the image to new positions.

When it comes to editing textures, brushes are a common tool. 3D painting systems apply them to draw a texture image directly on a surface (e.g. [10]). While this may include the creation or implicit modification of a parametrization [12], the focus of our work lies on the explicit modification of the texture map by the brushes. There is relatively few work in this context, since subsequent modifications to the textures are usually performed in the image space.

As our technique also features the use of positional constraints it is related to the subject of constrained texture mapping (see e.g. [15, 13]). However, the focus of our approach is on the local editing of the texture map using the geometry brushes and we do therefore not intent to compete with state-of-the-art in this regard.

More closely related is the work on texture mapping by Seo and Cordier [24] which allows to additionally constrain the gradients of the texture map in order to rotate and scale parts of the texture. They compute the warping using a time-dependant vector field similar to the shape deformation [6].

2 The Modeling Framework

Based on discrete Hodge theory, we formulate a method for integrating a given differential in the least squares sense on a simplicial surface (Sect. 2.1). We then apply this integration-module for modeling and propose a deformation framework (Sect. 2.2). Geometry brushes are defined in section 2.3 which deform the surface itself by specifying local deformations. Similarly, texture brushes for modeling the texture map are defined in section 2.4.

2.1 Integration Module

Deformation. The presented deformation framework starts from a triangular surface M represented by a set of $|V|$ vertices $p_i \in V \subset \mathbb{R}^3$, $i \in \{0, \dots, |V|-1\}$ and $|T|$ triangles t_j, $j \in \{0, \dots, |T|-1\}$. A deformation of the surface is a map $f_M : M \to \mathbb{R}^3$ which provides an embedding of M, i.e. new vertex coordinates. The undeformed surface corresponds to $f_M = \mathrm{Id}_M$.

Similarly, we consider deformations of a given texture map $\varphi : M \to \mathbb{R}^2$ on the surface. The deformed texture map is $f_\varphi : M \to \mathbb{R}^2$ where the undeformed state is represented by $f_\varphi = \varphi$. Note that the representation of the undeformed state is different than for surface deformations, since f_φ denotes the deformed state whereas f_M denotes the deformation between two surfaces.

In order to generalize both cases, we define a general *deformation map* $f : M \to \mathbb{R}^k$, $k \in \mathbb{N}$ with an undeformed state $f^* : M \to \mathbb{R}^k$. This can be a surface deformation, a texture deformation or a deformation on any scalar- or vector-valued function defined on M. We assume, that $f \in S_h^k$ where S_h is the space of (continuous) piecewise linear scalar maps on M. Any function $f \in S_h^k$ is represented by its values $f(p_i)$ at all vertices.

Differential. For any piecewise linear function $f : M \to \mathbb{R}^k$, the differential $Df : TM \to \mathbb{R}^k$ maps the tangent plane T_pM of any point $p \in M$ (from inside a triangle) into $\mathbb{R}^k$. After choosing a basis of the tangent plane, the differential is represented by the *Jacobian matrix* of dimension $k \times 2$:

$$D_p f(v) = \begin{pmatrix} \nabla f_0 \\ \dots \\ \nabla f_{k-1} \end{pmatrix} \cdot v, \forall v \in T_pM \tag{1}$$

where ∇f_i, $i \in \{0, \dots, k–1\}$ are the gradient fields (row-vectors) of the components of $f = (f_0, f_1, \dots, f_{k-1})$.

Comparing this matrix in different tangent spaces is not intuitive since they first have to be converted to the same coordinate system. Thus, for simplification, we extend the differential to the surrounding space $T_pM \times \mathbb{R} = \mathbb{R}^3$ and write it in Euclidean coordinates. The gradient vectors ∇f_i in Eqn. (1) remain the same, but are now represented in 3D-Euclidean coordinates. The dimension of the matrix becomes $k \times 3$, but is still of rank 2. Since f_i is piecewise linear, the gradient field is constant per triangle. It is computed on each triangle $t = (p_0, p_1, p_2)$ by (see e.g. [20]):

$$\nabla f_i(t) = \sum_{k=0}^{2} \cot(\alpha_k) \bar{f}_k \bar{p}_k, \tag{2}$$

where α_k are the inner angles at p_k and $\bar{p}_k := p_{(k+2) \bmod 3} - p_{(k+1) \bmod 3}$ and $\bar{f}_k := f_{(k+2) \bmod 3} - f_{(k+1) \bmod 3}$.

Integration. Our aim is to reconstruct a function f from a given differential

$$A = \begin{pmatrix} A_0^T \\ \dots \\ A_{k-1}^T \end{pmatrix}, \quad A_0, \dots, A_{k-1} : M \to R^3 \tag{3}$$

such that $Df = A$. In general, such a map does not exist since A must satisfy some integrability conditions. The vectors A_j, $j \in \{0, \dots, k-1\}$ must be gradient vector fields which can be integrated globally to a scalar function.

This setting is strongly related to Hodge-Helmholtz decomposition [9]. Given a differentiable two-dimensional manifold M, any arbitrary vector field X on M can be uniquely decomposed into the sum of three vector fields

$$X = \nabla u + \mathrm{J}\nabla v + H, \tag{4}$$

with a *potential* $u : M \to \mathbb{R}$, a *co-potential* $v : M \to \mathbb{R}$ and a harmonic vector field H. J denotes the rotation of all vectors by 90 degrees in their local tangent plane.

When applying the Hodge-Helmholtz decomposition to the row-vectors of any differential Df, the co-potential and harmonic parts vanish, since the row vectors are already gradient fields. The Hodge-Helmholtz decomposition separates any given $k \times 3$ matrix A into a differential and a non-integrable part (the co-potential and harmonic part).

The potential can be computed by projecting the given matrix A into the space of Jacobi matrices in L_2 sense, i.e. by minimizing:

$$E(f) = \int_M \|Df - A\|^2 \tag{5}$$

for $f : M \to \mathbb{R}^k$. Deriving the Euler-Lagrange equation results in a Poisson equation:

$$\Delta f = \operatorname{div}(A), \tag{6}$$

where $\operatorname{div}(A)$ is a vector whose entries $\operatorname{div}(A)_i$ contain the discrete divergence of the vector field formed by the i-th row of A.

In the discrete case, where M is a triangular surface, there is an analogue theory, called the discrete Hodge-Helmholtz decomposition [22]. Eqn. (4) is similar, but the function spaces are: $u \in S_h$, $v \in S_h^*$ and H is a discrete harmonic vector field. S_h^* is the space of non-conforming finite element functions which are defined by values on edge midpoints.

The potential is found by minimizing

$$E(f) = \sum_{t \in T} \operatorname{area}(t) \|Df_{|t} - A_{|t}\|^2 \tag{7}$$

for $f \in S_h$. The minimizer is found by setting all derivatives of this energy to 0 which leads to k systems of linear equations:

$$L\mathbf{f}_j = b_j, \quad j \in \{0, \dots, k-1\} \tag{8}$$

where L is the *cotan-Laplace* matrix. When interpreting the j-th column of A as a vector field A_j, then $b_j \in \mathbb{R}^{|V|}$ denotes the divergence vector which contains the divergence of A_j at all vertices (see [22], Eqn. (7) and [28], Section 2.4.4). The solution vector $\mathbf{f}_j \in \mathbb{R}^{|V|}$ contains the values of the j-th component of f at all vertices.

2.2 Deformation Framework

The integration-method described in Sect. 2.1 allows to generate a surface from an arbitrary user-given differential A. This module is the foundation for the proposed modeling tools.

Starting from a given *domain surface* M the user applies several deformations. The actual deformed surface is denoted by M'. Whenever another deformation is applied to M', we first compute the differential of the current embedding $f : M \to M'$ (resp. of the current texture map $f : M \to \mathbb{R}^2$). Then, the user-interaction is translated into local modifications $A^{mod}(A)$ of the differential A in each point of M. Finally, the integration-module is called which assures a globally consistent differential and generates the deformed surface (resp. texture map).

Since the domain surface M never changes, the matrix L from Eqn. (8) always remains the same and can be factorized once at the beginning. Every call of the integration-module just solves $k = 3$ (resp. $k = 2$ for texture editing) many systems of equations with different right vectors. Solving this linear system of equations can be done very fast, so this technique allows for applying deformations in real-time. A summary of the deformation process is given by Algorithm 1.

Algorithm 1: Interactive Deformation

```
Input: triangular surface M, possibly with texture
       coordinates (u, v)
Assemble matrix L and compute pre-factorization;
Initialize deformation f := Id, resp. f := (u, v);
while True do
    Compute differential A := Df;
    Modify A into A^mod, e.g. by user interaction;
    for i = 0 to k − 1 do
        Compute divergence b_j := div(A_j^mod);
        Solve L f_j^mod = b_j;
    end
    Set f := f^mod and re-draw surface;
end
```

In Sect. 2.3 and 2.4, we propose common geometry and texture brushes. Each brush defines a deformation of the differential and can be applied as follows:

Global deformation: Apply the deformation to the whole surface.

Semi-local deformation: Apply the deformation to a set of selected triangles. This is useful if specific parts of the surface need to be transformed.

Local deformation: Apply the deformation to all triangles in the vicinity of a picked point on the surface. After choosing a radius of influence, the user paints the deformation via click and drag onto the surface.

For local and semi-local deformations, we also propose the use of *fading brushes* which apply a large deformation to the center and fade out to the boundary

of the region of interest. Fading brushes often provide a smoother transition for strong deformations.

2.3 Modeling Brushes

The number of possible modeling brushes one can think of is almost unlimited. Here, we introduce the most canonical ones. It is quite simple to extend our framework by further user-defined brushes.

Isotropic Scale Brush. This brush scales the area by a given factor $\lambda \in \mathbb{R}$, i.e. the modified differentials as used in Sect. 2.2 are $A^{mod}(A) = \lambda \cdot A$. The brush can be used to shrink or enlarge specific parts of the surface uniformly, as shown in Fig. 1.

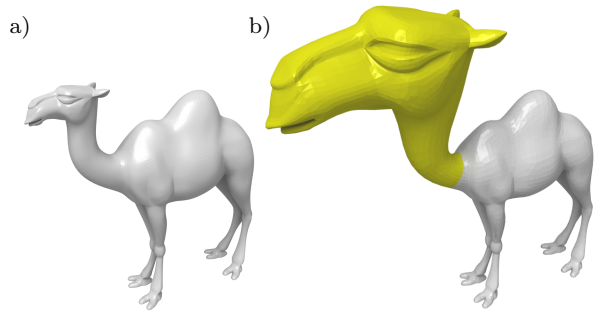


Fig. 1 a) Undeformed surface. b) Deformation using the scale brush.

Anisotropic Scale Brush. Given a vector field with vectors of unit length $v_p \in S^3$ and a scaling factor $\lambda_p \in \mathbb{R}$. The modified differentials are

$$A^{mod}_{v,\lambda}(A) = A + (\mathrm{Id} + (\lambda_p - 1)v_p v_p^T) \tag{9}$$

which corresponds to an anisotropic scaling in direction of v_p by a factor of λ_p.

If λ_p and v_p are chosen to be constant everywhere, the shape is *thickened* anisotropically in the direction of v_p (see Fig. 2, top).

Alternatively, v_p can be defined as function on M, e.g. as one of the principal curvature directions. Scaling in minimum principal curvature direction elongates cylindrical parts of the surface while the maximum principal curvature direction thickens these parts (see Fig. 2, bottom).

Rotate Brush. This deformation multiplies the differentials with a rotation matrix R, i.e. the modified differentials are $A^{mod}(A) = R \cdot A$. The user first specifies a rotation axis. While clicking and dragging on the surface, the rotation is applied to all triangles in the chosen area (see Fig. 3).

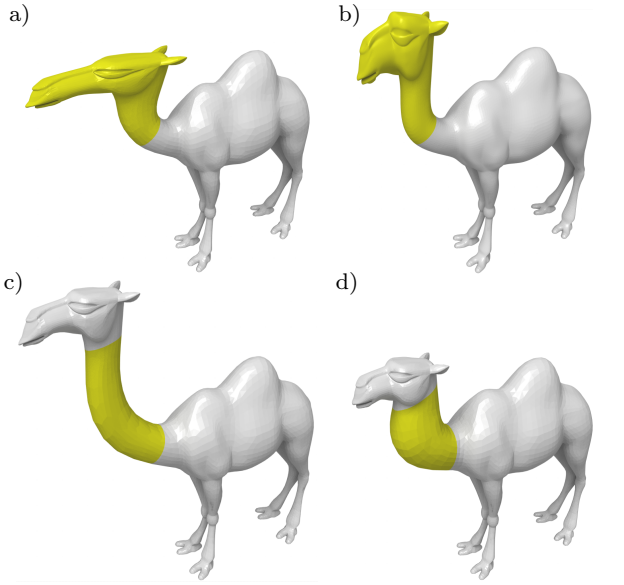


Fig. 2 Anisotropic scale a) in x-direction, b) in z-direction, c) in minimum principal curvature direction, d) in maximum principal curvature direction.

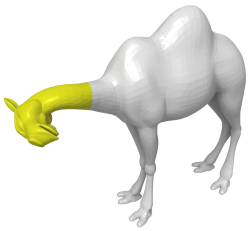


Fig. 3 Rotate Brush.

Bump Brush. The bump brush imprints a hill or bump in normal direction onto the surface (see Fig. 4).

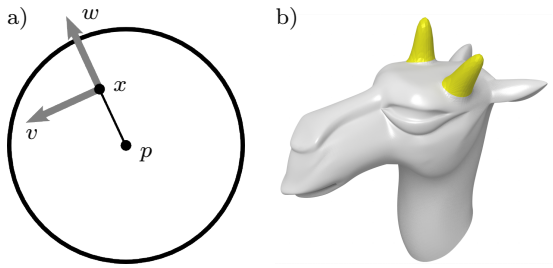


Fig. 4 Application of the bump brush. a) Schematic view of the region of interest. b) Create bumps with a single click.

Given a center point $p \in M$, a radius $r > 0$ and a slope value $m \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. The differential in each point x within a disk of radius r around p is transformed by (see Fig. 4, top left):

1. Rotation around an axis v by an angle α:

$$v := N \times \frac{x-p}{\|x-p\|}, \quad \alpha := m \sin\left(\pi \frac{\|x-p\|}{r}\right) \tag{10}$$

   where N is the surface normal in p.
2. Scale anisotropically in the direction of w by a factor of λ:

$$w = \frac{x-p}{\|x-p\|}, \quad \lambda = \frac{1}{\cos\alpha}. \tag{11}$$

Choosing a slope of $m > 0$ adds a bump onto the shape while negative slope produces a valley.

Constraints. We want to note that our framework supports hard constraints, similar to most other deformation techniques. A set of vertices can be forced to a desired position by constraining their values when solving the system of equations given by Eqn. (8). Both, hard constraints and local control over the differential, can be used simultaneously.

2.4 Texture Brushes

A texture brush modifies a 2D texture map on the surface. It is applied similarly to a modeling brush by clicking or dragging on the surface or directly to a selected region.

Texture Scale Brush. The anisotropic or isotropic texture scale brush works similarly to the corresponding modeling brush. The modified differentials are $A^{mod}(A) = \lambda A, \lambda \in \mathbb{R}$. The metric of the texture map is locally scaled up or down (Fig. 5).

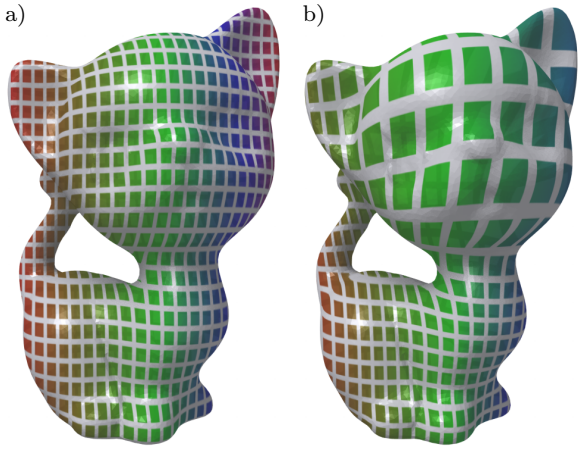


Fig. 5 Texture scale brush. a) Input texture. b) Deformed texture.

Texture Rotate Brush. Applies a rotation in the tangent plane to the differentials of the texture map, see Fig. 6.

Fig. 6 Texture rotate brush.

Texture Constraints. Using hard constraints for texture deformation, we can adjust a texture image by defining special feature points on the surface and their corresponding positions in the texture image. The integration-module is called on the undeformed differential and the given constraints. Texture constraints can always be specified, also in addition to the use of texture deformation brushes.

2.5 Implementation Details

We have implemented our modeling framework in Java using the TAUCS library [27] to decompose the Laplace matrix L and solve the system of equations (8). Since the matrix is of co-rank 1, a Sparse Cholesky Decomposition (SCD) cannot be applied directly.

One way to handle this problem would be to eliminate one variable from the system. We found it easier though to add a sufficient small positive number ϵ on the diagonal of the matrix, so it becomes positive definite and an SCD can be applied.

3 Results and Applications

We tested our technique on various models. The maximum computation time on all our tested models consisting of several ten-thousand triangles is about a few hundred milliseconds if the system is completely unconstrained. However, when editing a specific area the user may fix other parts of the model thus eliminating the according variables from the system of equations and significantly reducing the computation time with the additional advantage that the fixed parts will be perfectly preserved.

The distinctive feature of our modeling framework is the use of geometry brushes. A first impression of modeling with the geometry bushes is provided by Fig. 7. One can see, that enlarging the nose of the Caesar model in this way is very intuitive. In fact it takes only a few seconds. To see how the brushes are used to perform a more complex editing task and to get a further impression of the different brushes the reader may also consult the accompanying video.

Brushes are very intuitive and provide the artist with a fundamentally different way of interacting with the geometry. While in many situations other metaphors may be more effective, brushes prove to be especially useful in those scenarios where the other approaches are impossible or at best tedious to apply. Therefore, geometry brushes are a supplement rather than a competitor to the present set of interaction tools.

On this basis, it is difficult to conduct a fair comparison with related work that applies other modeling metaphors. As Sect. 1.1 highlights, there are methods that use brushes to perform deformation but most of them are too different to enable a reasonable comparison. The most interesting tool to compare with in terms of modeling would certainly be ZBrush but there is unfortunately no scientific literature available on the subject. With regard to the local editing of texture maps the present alternatives are even less suitable for comparison, since most related work in this area is focussed on the use of positional constraints.

For those reasons we have decided to stay clear of any comparison but instead provide examples of editing tasks that can potentially benefit from the use of geometry brushes. Local deformations using geometry and texture brushes are shown in Sect. 3.1. A strength of the framework is the possibility to define custom brushes very easily and intuitively. Sect. 3.2 shows example brushes for different applications.

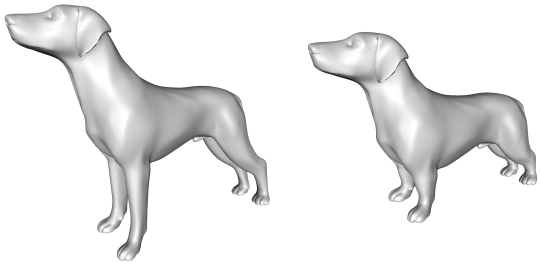


Fig. 8 Left: undeformed surface. Right: deformed surface using the scale brush to shrink neck and legs.

Fig. 9 Left: undeformed model. Right: applying rotation with linearly varying rotation angle.

3.1 Application of Brushes

The following examples show results of using geometry brushes. The anisotropic scale brush is especially suited to shrink cylindrical areas, such as the neck or legs of animals, see Fig. 8. This deformation is generated by first choosing a stroke size and then by painting on the neck and legs until the desired shape is reached.

Fig. 9 shows a rotation about the height axis. We have chosen a fading brush whose stroke size is the diameter of the geometry. Therefore, the amount of rotation varies linearly from α on top to 0 on the bottom.

The octopus in Fig. 10 is modeled by applying rotations whose angle varies linearly along the longitudinal direction. Similar to the buddha, we just applied a fading rotate brush to the tentacles.

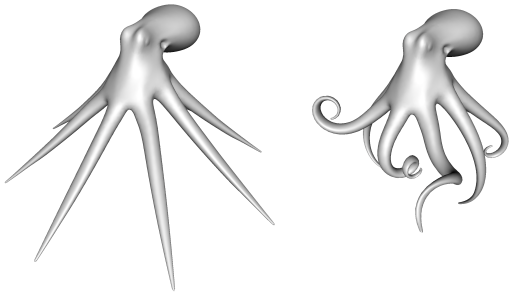


Fig. 10 Left: undeformed model. Right: applying rotate brush.

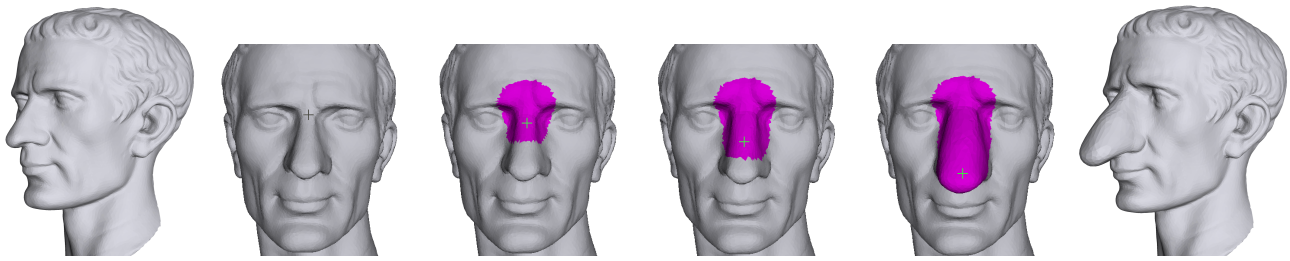


Fig. 7 Editing with brushes. The nose of the Caesar model (left) is enlarged in real-time by a single continuous stroke with a fading scale brush (middle images, parts affected by the brush coloured in magenta). The result can be seen to the right.

3.2 Custom Deformations

Beside the proposed brushes, it is quite simple to define custom brushes with specific behaviour for the actual application. Here we show some examples of custom geometry and texture deformations. A custom deformation is relatively simple to implement, once known how the differential must be deformed locally.

Rotate Planes. As a theoretic experiment, we used the differential-based deformation technique to rotate all tangent planes by a constant angle around the surface normals. It is similar to the rotate brush, but here, the rotation axis varies over the geometry.

For most surfaces, the global rotation of the tangent planes makes no geometric sense, except for minimal surfaces. Each minimal surface M comes with an *Associated Family* of minimal surfaces M_α, $\alpha \in \mathbb{R}$ where each surface of the family has the same Gauss map g [3]. The normal vectors in every point of M and the corresponding point in M_α are identical, and the tangent planes are locally rotated by α.

Fig. 11 shows the results on applying this deformation to minimal surfaces. Notice, how the conjugate minimal surfaces (those for $\alpha = \pi/2$) are nicely reconstructed. We are aware that this kind of deformation might not have many applications for surface modeling. We rather see this feature as a theoretical justification of our algorithm.

Cartooning. This custom deformation scales all tangent planes locally dependent on the curvature tensor at each point. Let $p \in M$ be a point on the surface and e_1, e_2 the principal curvature directions in p with curvature values κ_1, κ_2. For a given $\lambda \in \mathbb{R}$, we apply two anisotropic scalings in direction of e_i, $i = 1, 2$:

$$\left(\frac{|\kappa_i|}{|\kappa_{\max}|} \mathrm{Id} + (1 - \frac{|\kappa_i|}{|\kappa_{\max}|}) S(e_i, \lambda) \right), \tag{12}$$

where S is the anisotropic scale matrix from Eqn. (9) and $\kappa_{\max}$ is the absolute maximum curvature value of all κ_1 and κ_2 on M.

Areas with high curvature are maintained while flat or cylindrical areas are scaled by λ. The effect is global and scales the whole surface. We rescale the surface after each application uniformly to its initial dimension, thus it appears that changes are only made to areas of high curvature. For $\lambda > 0$, areas of high curvature are shrinked and surface features are sharpened. For $\lambda < 0$ highly curved regions are blown up and generate a cartoon-like effect (see Fig. 12).

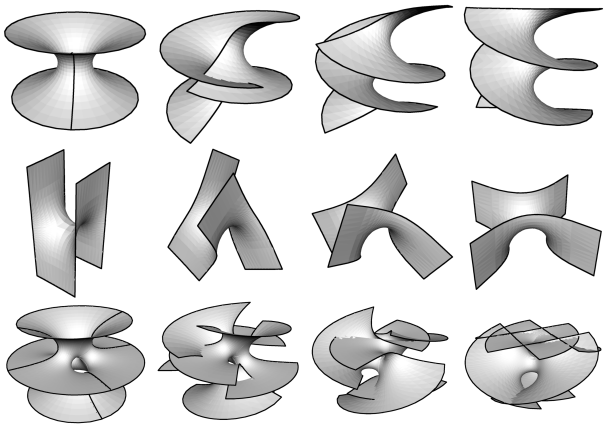


Fig. 11 Global Rotation of all tangent planes on minimal surfaces produces the Associated Family. Rotations are done by 0, 30, 60 and and 90 degrees, where the last model shows the conjugate surface. Top: catenoid is transformed into the helicoid, Middle: transformation between first and second surface of Scherk, Bottom: Costa Surface.

While conceptually similar, the intention and effect of our approach are different to the curvature clamping and feature enhancement of [5] as illustrated by Fig. 13. The difference is particularly apparent at the ears. They are not only smoothed (a) resp. sharpened (c) but also overproportionally contracted resp. expanded in the direction of less curvature. Note further, that our method has advantages in terms of performance since the processing by [5] involves a non-linear optimization.

Texture Deformation using Constraints. Hard constraints for texture deformation allow alignment of the texture image by moving *handles* on the surface, like in 2D image deformation techniques.

Fig. 12 Cartooning of the Caesar head. Middle: Undeformed model. Left: Smoothen features ($\lambda < 0$). Right: Sharpen features ($\lambda > 0$).

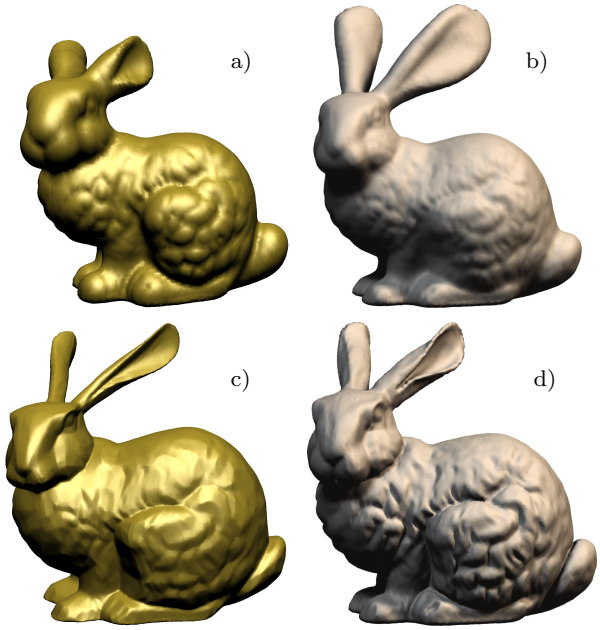


Fig. 13 Comparison of cartooning (left) with the approach of [5] (right). Top: Smoothen features ($\lambda < 0$), curvature clamping. Bottom: Sharpen features ($\lambda > 0$), feature enhancement.

Fig. 14 shows an example of texture deformation by specifying single points as constraints. Adjusting a texture image with this technique is simple and intuitive.

We provide this example to show how positional constraints may be incorporated in our framework. Nevertheless, the focus of our work lies on the direct manipulation of texture gradients using the brushes as already noted in Sect. 1.1 and we do not provide any comparison with similar methods for that reason.

Magnifying Brush. Fig. 15 shows an example of a very specific brush for design purposes. It is applied to the texture map and magnifies the selected area. At the same time, it is compressed near the border such that the texture map outside remains nearly unaffected.

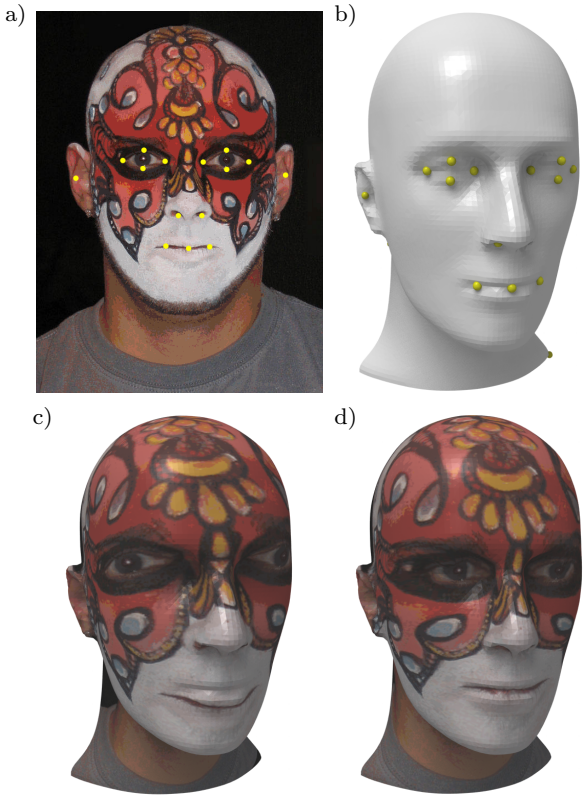


Fig. 14 Specify hard constraints on texture coordinates. a) Texture image with handles drawn in yellow. b) Target geometry. Yellow handles are placed manually. c) Initial texture layout by projection. d) Deformed texture layout respecting the constraints from the handles.

Using the notation of Fig. 4, we define $d = \frac{\|x-p\|}{r}$ and apply the anisotropic scales $S(w, \lambda_w)$ and $S(v, \lambda_v)$ to the differentials with:

$$\lambda_w = \frac{1}{\frac{\sqrt{d}}{\lambda} + \lambda(1-\sqrt{d})}, \quad \lambda_v = \frac{1}{\frac{d}{\lambda} + \lambda(1-d)}, \tag{13}$$

where $\lambda \in \mathbb{R}$ defines the amount of magnifying.

4 Conclusion

The deformation approach proposed in this article is based on a simple and flexible technique which assures that any local modifications produce a globally consistent deformation.

This technique was utilized in a novel way by combining it with geometry brushes to form a robust and interactive modeling framework. By extending this framework for editing textures directly on a given geometry, the user is provided with an effective way to adjust textures to his needs without constantly having to switch between geometry and texture view.

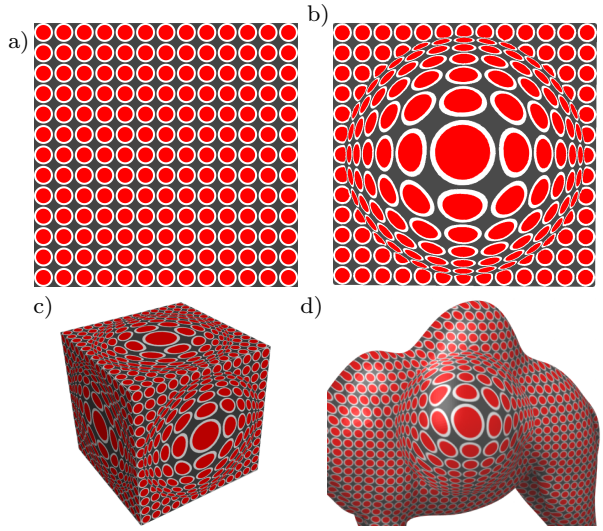


Fig. 15 a) Planar surface with texture image. b) Application of the magnifying brush. c) and d) Magnifying textures applied to surfaces.

For both, textures and shapes, the proposed brushes allow for intuitive editing and provide added value to the user. Therefore, they are a useful supplement to the already great variety of editing tools available.

References

1. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Transactions on Visualization and Computer Graphics **14**, 213–230 (2008)
2. Crane, K., Pinkall, U., Schröder, P.: Spin transformations of discrete surfaces. ACM Trans. Graph. **30**(4), 104:1–104:10 (2011)
3. Dierkes, U., Hildebrandt, S., Küster, A., Wohlrab, O.: Minimal Surfaces, vol. 1. Springer (1992)
4. Eigensatz, M., Pauly, M.: Positional, metric, and curvature control for constraint-based surface deformation. Comput. Graph. Forum **28**(2), 551–558 (2009)
5. Eigensatz, M., Sumner, R.W., Pauly, M.: Curvature-domain shape processing. Comput. Graph. Forum **27**(2), 241–250 (2008)
6. von Funck, W., Theisel, H., Seidel, H.P.: Vector field based shape deformations. ACM Trans. Graph. **25**, 1118–1125 (2006)
7. Gal, R., Sorkine, O., Cohen-Or, D.: Feature-aware texturing. In: Proceedings of Eurographics Symposium on Rendering, pp. 297–303 (2006)
8. Gal, R., Sorkine, O., Mitra, N., Cohen-Or, D.: iwires: An analyze-and-edit approach to shape manipulation. ACM Trans. Graph. **28**(3), 33:1–33:10 (2009)
9. Girault, V., Raviart, P.: Finite element methods for Navier-Stokes equations. Springer, Verlag Berlin (1986)
10. Hanrahan, P., Haeberli, P.: Direct wysiwyg painting and texturing on 3d shapes. SIGGRAPH Comput. Graph. **24**(4), 215–223 (1990)
11. Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y.: Subspace gradient domain mesh deformation. ACM Trans. Graph. **25**, 1126–1134 (2006)
12. Igarashi, T., Cosgrove, D.: Adaptive unwrapping for interactive texture painting. In: Proceedings of the 2001 symposium on interactive 3D graphics, I3D '01, pp. 209–216. ACM, New York, NY, USA (2001)
13. Kraevoy, V., Sheffer, A., Gotsman, C.: Matchmaker: constructing constrained texture maps. ACM Trans. Graph. **22**(3), 326–333 (2003)
14. Lawrence, J., Funkhouser, T.: A painting interface for interactive surface deformations. Graph. Models **66**, 418–438 (2004)
15. Lévy, B.: Constrained texture mapping for polygonal meshes. In: Siggraph, pp. 417–424 (2001)
16. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. **24**(3), 479–487 (2005)
17. Milliron, T., Jensen, R.J., Barzel, R., Finkelstein, A.: A framework for geometric warps and deformations. ACM Trans. Graph. **21**, 20–51 (2002)
18. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Fibermesh: designing freeform surfaces with 3d curves. ACM Trans. Graph. **26**(3) (2007)
19. Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D.: A sketch-based interface for detail-preserving mesh editing. ACM Trans. Graph. **24**(3), 1142–1147 (2005)
20. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. J. Exp. Math. **2**(1), 1–28 (1993)
21. Pixologic, Inc.: Zbrush. URL `http://www.pixologic.com/zbrush/`
22. Polthier, K., Preuss, E.: Identifying vector field singularities using a discrete Hodge decomposition. In: Visualization and Mathematics III, pp. 113–134. Springer (2003)
23. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Trans. Graph. **25** (2006)
24. Seo, H., Cordier, F.: Constrained texture mapping using image warping. Comput. Graph. Forum **29**(1), 160–174 (2010)
25. Sorkine, O., Botsch, M.: Tutorial: Interactive shape modeling and deformation. In: Eurographics (2009)
26. Takayama, K., Schmidt, R., Singh, K., Igarashi, T., Boubekeur, T., Sorkine, O.: Geobrush: Interactive mesh geometry cloning. Comput. Graph. Forum **30**(2), 613–622 (2011)
27. Toledo, S.: Taucs: A library of sparse linear solvers, version 2.2. (2003). URL `http://www.tau.ac.il/~stoledo/taucs/`
28. Wardetzky, M.: Discrete differential operators on polyhedral surfaces - convergence and approximation. Ph.D. thesis, Freie Universität Berlin (2006)
29. Xu, W., Zhou, K.: Gradient domain mesh deformation – a survey. J. Comput. Sci. Technol. **24**(1), 6–18 (2009)
30. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. **23**(3), 644–651 (2004)
31. Zhou, Q., Weinkauf, T., Sorkine, O.: Feature-based mesh editing. In: Proc. Eurographics, Short Papers (2011)
32. Zimmermann, J., Nealen, A., Alexa, M.: Silsketch: automated sketch-based editing of surface meshes. In: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling, SBIM '07, pp. 23–30. ACM, New York, NY, USA (2007)