Computers & Graphics 35 (2011) 614–622

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SMI 2011: Full Paper

Freestyle: Sculpting meshes with self-adaptive topology

Lucian Stănculescu [a,b,c,*], Raphaëlle Chaine [a,b], Marie-Paule Cani [c,d,e]

[a] Université de Lyon, CNRS, France
[b] Université Lyon 1, LIRIS, France
[c] LJK, Grenoble, France
[d] Université de Grenoble, France
[e] INRIA, France

ARTICLE INFO

Article history:
Received 12 December 2010
Received in revised form
14 March 2011
Accepted 14 March 2011
Available online 22 March 2011

Keywords:
Adaptive topology
Interactive techniques
Quasi-uniform mesh

ABSTRACT

We present a real-time method for sculpting triangular manifold meshes while enabling arbitrary surface deformation with seamless topological changes. Our insight is that the use of quasi-uniform mesh sampling, an interesting option now that very large meshes can be edited and displayed in real-time, provides the right framework for expressing and efficiently processing arbitrary changes of topological genus. The user controls deformation by gesture: he sweeps tools that apply a variety of deformation fields, from smoothing and trimming ones to local inflation and constant volume deformation tools. Meanwhile, the quasi-regular mesh seamlessly splits or locally blends when and where needed, while still following the user-specified deformation. Our method guarantees a closed, self-intersection-free mesh, whatever the user action. We demonstrate the practical usability of the resulting, interactive sculpting system through the sculpture of models that would have been extremely difficult to achieve with both current research methods and state of the art professional software.

1. Introduction

Digital sculpting has become one of the most important tools for 3D content creation in the film industry, special effects and computer games, slowly replacing more traditional methods that use individually controlled points to adjust surface patches or create a simple mesh that will be later globally subdivided. The introduction and rapid development of sculpting applications has immensely increased the workflow of professional artists and even attracted a number of amateur users to digital media. However, few of the existing software enable topological changes such as local splitting or merging of model parts. More precisely, on one hand we find polygonal methods [14,2,3] that impose a fixed mesh connectivity and topological genus during sculpting. This allows them to store mesh connectivity and vertex positions directly on the GPU while editing positions only in deformed regions. Attempts to introduce adaptive mesh refinement in this framework lead to a relative drop in performance [12] while the topology of the model still has to be predefined before starting to sculpt. On the other hand, grid-based methods [13] handle dynamic changes of topological genus: they sample polygonal objects into regular voxel grids in which material can be locally added or removed. Unfortunately, mesh quality can be spoiled by these successive re-samplings and several of the advantages of meshes, such as their ability to carry texture or to maintain surface details during larger scale deformation, are lost.

This work presents an effective strategy for enhancing mesh-based sculpting methods with intuitive topological changes. The latter take place seamlessly while the user sweeps tools to control deformation: the model eventually splits into pieces where it becomes very thin, while two parts that come close enough automatically blend. The method insures that the mesh always remains composed of closed connected components and is free of self-intersection, while also favoring triangles with good aspect ratio suitable for quality rendering. Let us first present a quick overview of prior work before introducing our contributions.

1.1. Related work

The professional applications we already listed were inspired by a quite impressive amount of research work on interactive sculpting in the last 20 years:

Introduced in the early nineties by the seminal work of Galeyan and Hughes [10], grid-based methods allow the user to interactively add or remove material, stored as a density field in a voxel grid. The objects surface is defined as an iso-surface of this density, and converted into a triangular mesh using marching-cubes. These methods were enhanced by the introduction of local deformation tools and resolution adaptive grids such as those by Ferley et al. [8,9]. Constant volume, large scale deformations were

* Corresponding author at: Université de Lyon, CNRS, France.
Tel.: +33 950440515.
E-mail address: lucian.stanculescu@gmail.com (L. Stănculescu).

0097-8493/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.03.033

investigated too, making the resulting model very similar to some virtual clay like the one developed by Dewaele et al. [7]. These methods allow the model to freely change topological genus during edition, at the scale of the local grid resolution. However, as the surface is extracted from the grid, triangles are recomputed each time a part of the model moves, even when the amount of local deformation is small such as when an object is twisted of bent. This can cause details to be blurred by the successive re-samplings into the grid. Moreover, since the triangles used for display are not persistent during the editing process, attaching properties to the surface such as bumps or texture is difficult.

Other implicit representations, such as the Blob Tree framework, have also been successfully used in sculpting frameworks. Here, constructive methods such as [16] allow to create heterogeneous objects with adaptive topology, while WarpCurves [19] allow to deform implicit surfaces by using curves drawn on the surface. The main inconvenient of the Blob Tree is that applying many tools in a certain region can slow down the local extraction of the surface, since everything is stored in a hierarchy which increases with each change. Using grids to store the values of the implicit function overcome this inconvenient, but brings us back to grid-based methods.

Another strategy for sculpting a model is to apply deformations directly to mesh vertices, and possibly combine the process with some adaptive mesh refinement to better capture details creation. Two kinds of approaches were used to do so: model-based deformations such as Laplacian editing [18] use the triangle mesh to compute local geometric features, which are best preserved during larger scale deformations; in contrast, space deformations enforce mesh vertices to follow a deformation field specified everywhere in space, independently from the models geometry. These space deformations can be driven by sweeping gestures and were later extended to constant volume deformations such as the swirling sweepers by Angelidis et al. [1] or the more general approach by von Funck et al. [21]. Although adaptive mesh refinement can be incorporated into these methods, no change of topological genus is permitted, since space deformations are required to be fold-over free to prevent mesh self-intersection.

Our method belongs to this class of space deformation methods, which enables us to easily provide constant volume sculpting tools. However, similar to grid-based methods, it seamlessly handles changes of topological genus such as splits and merges, at a pre-defined resolution scale.

Although never provided in previous sculpting systems, temporally coherent meshes that track a surface which splits and merges over-time were already proposed in the area of data segmentation and tracking. The key idea is to extract the mesh from a volumetric tetrahedrization [15] or to use an additional particle system to adaptively sample the input data [6]. The mesh connectivity is mainly changing in regions that undergo topological changes. However, the tetrahedrization update and the relaxation framework of the interacting particle system are a bottleneck that prevent their use in interactive sculpting applications. The approach we develop is closer to the method of Lachaud et al. [11], which uses an evolving closed mesh to segment some static volumetric data defined on a fixed voxel grid. The authors want this mesh to be uniform and perform changes of topological genus whenever two different parts of the surface get close to each other. However, in contrast to ours, the method does not ensure that the desired uniformity properties of the mesh are preserved.

1.2. Overview

We introduce the first real-time sculpting system – to our knowledge – that combines a mesh-only representation with seamless changes of topological genus. As in grid-based methods, topology is controlled from a single user-specified resolution threshold: any model part thinner than this threshold automatically splits; two parts of the mesh that come to a smaller distance automatically merge. The key feature of the method is to rely on a restricted mesh structure, which we call *quasi-uniform mesh*, to represent geometry. When deformed, this quasi-uniform mesh automatically splits and merges when and where needed, leaving the user concentrate on the shape he or she is creating. We demonstrate the practical usability of this method by implementing it within an interactive sculpting system, where the user interacts through a wide range of gesture-based space deformation tools. As our results show, this system has already been successfully used by a professional artist.

The remainder of the paper is organized as follows: Section 2 formally defines quasi-uniform meshes and discusses their properties. Time evolution under a deformation field is described in Section 3. The use of this method in an interactive sculpting system is described in Section 4. Section 5 presents results and discusses the benefits and limitations of our rather simple quasi-uniform structure compared with previous adaptive or multi-resolution representations. We conclude and describe future work in Section 6.

2. A nearly uniform framework

For our method we need an underlying geometric model that is simple, automatic and able to adapt to a variety of deformations at interactive frame rates, based on some intuitive properties that are kept throughout the sculpting process.

Our method is restricted for now to triangular manifold meshes, so, in the following, a mesh is a triangular manifold mesh unless otherwise specified.

Observing that to maintain an adaptive sampling of a surface can be costly, especially when topological genus changes arise, we propose to focus on simpler and nearly uniform meshes offering the following benefits:

- Connectivity and sampling of the vertices evolve dynamically, without resorting to any relaxation process, while ensuring some tight, nearly uniform distribution of the vertices on the surface and promoting a quality triangulation.
- Nearly uniform sampling can be exploited to help the detection and the handling of topological genus changes.
- The main parameter of the data structure can be given a physical meaning with regard to the material the object is made of, namely the level of detail that it can yield when being sculpted. We also add a second parameter in Section 3 reflecting the thinning that this material can withstand before breaking locally.

For this, we define two categories of manifold meshes: d_{detail} *tight* meshes and quasi-uniform meshes. The first ones ensure a tight sampling of the surface, while the second also favors a uniform vertex distribution over the surface. We also show how to generate such structures.

2.1. d_{detail} tight mesh property

Definition 2.1. Given a closed manifold mesh M and a threshold d_{detail}, M is said to be a d_{detail} tight mesh if every edge in M is smaller than d_{detail}.

The advantage of a d_{detail} tight mesh is that it has enough vertices to reflect the geometry of the underlying surface with a precision better than d_{detail}, so the use of mesh connectivity is no

longer needed when making localization tests at a resolution less accurate than d_{detail}.

Lemma 2.1. *Given a point P and a d_{detail} tight mesh M, if the distance between P and M is smaller than $\sqrt{2/3}d_{detail}$, there exists a vertex V of M whose distance to P is also smaller than d_{detail}.*

Proof. The minimum distance between a point P and a d_{detail} tight mesh is greater than $\sqrt{2/3}d_{detail}$ if the distance from P to the mesh vertices exceeds d_{detail}. This minimum distance is obtained when P forms a regular tetrahedron with the three vertices of a triangular facet. □

We have proven than the localization – finding the distance – of a point P with respect to a tight mesh M can safely be approximated by that between the point P and the closest vertex of the mesh M.

2.2. Converting a manifold mesh to a d_{detail} tight mesh

A d_{detail} tight mesh can easily be obtained from any initial manifold mesh describing the surface by performing the following procedure, whose complexity is $O(e)$ where e denotes the number of edges in the resulting mesh.

Proceduce 2.1. *Given a manifold mesh M, the d_{detail} tightness property can be ensured by iterating over all the edges of M and splitting those which are larger than d_{detail}. The split operation amounts to adding a vertex at the midpoint of an edge and splitting the two neighboring triangles accordingly. The iteration over the edges is performed using a simple queue and the resulting new edges are inserted in the queue.*

The edge split operation is illustrated in Fig. 1.

Although interesting for us, the d_{detail} tight property is not sufficient to ensure good quality triangles. In the case where time is not an issue, such as in the initial preparation of a mesh for sculpting, we can also use a priority queue for better triangle quality. For further improvement and during the sculpting process we will use our quasi-uniform mesh structure, presented next, which also strives to promote a minimum length for the edges.

2.3. Quasi-uniform mesh structure

Definition 2.2. A manifold mesh is said to be compliant with a minimum length d, if it is obtained by iterating over all the edges of an initial mesh M while collapsing those whose length is smaller than d. If an edge collapse gives rise to intersecting triangles, then it is not performed.

Since an edge collapse moves its vertices towards the midpoint position of the disappearing edge (Fig. 2), we do not ensure that all the resulting edges will remain larger than d, although we still maintain this property for a large percentage of the mesh edges, or that they stay smaller than d_{detail}, so the tightness property can also be broken by performing the minimum length compliance procedure.

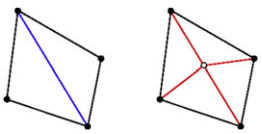

Fig. 1. Edge split. As the blue edge splits into two equal segments, the four created edges in red are smaller than the largest initial edge. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

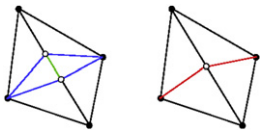

Fig. 2. Edge collapse. This operation is performed if the size of the green edge is smaller than d. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

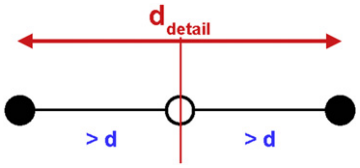

Fig. 3. Relation between d and d_{detail}.

Given a manifold mesh M, note that it is more reasonable to make it compliant with a minimum length d satisfying $d \leq d_{detail}/2$ before establishing the d_{detail} tight property on M. Indeed, this ensures that the edges of length greater than d_{detail} after the sequence of collapse operations will not be split into edges of length less than d (as illustrated in Fig. 3).

In practice, a value slightly smaller than $d_{detail}/2$ is optimal (see justification in Section 5).

Definition 2.3. A mesh M is said to be quasi-uniform if there exists d_{detail} and d with $d \leq d_{detail}/2$, such that M results from the compliance of an initial manifold mesh with a minimum edge length d, followed by the restoration of the d_{detail} tight property.

A quasi-uniform mesh is a mesh that ensures that all edges will be smaller than d_{detail} while favoring (without any guarantee) edge lengths to be larger than d. d can be seen as an internal parameter to the data structure, since it is bound to d_{detail}.

In the following, we will use and maintain quasi-uniform meshes to represent the geometry of our sculptures. The next section explains how these models are deformed and seamlessly change topology under the action of a deformation field controlled by the user.

3. Temporal evolution under deformation

Let us now demonstrate the interest of using a quasi-uniform mesh structure to track an arbitrary deformation field, supposed to be known with a resolution corresponding to d_{detail}. We will also show that the introduction of a new parameter $d_{thickness}$ combined with a consistent time sampling, ensures that the surface always remain manifold and does not self-intersect. The topological genus changes are then triggered by excessive thinning of the surface using simple topological operations.

3.1. Tracking a space deformation over time

Given a quasi-uniform mesh M and a deformation field D, we exploit the guaranteed tightness of the sampling, by considering the value of the deformation field only at the vertices of the mesh. This eliminates any need for interpolating the field in the surrounding space even when the field depends on characteristic properties at each vertex such as the normal. To define the maximum displacement step useful for tracking the movement of the evolving surface, we need to know the minimum thickness tolerated by the volume delimited by this surface. This thickness can be seen as a physical characteristic of the material in the

virtual sculpture application based on our data structure, presented in Section 4.

Definition 3.1. Let $d_{thickness}$ denote the minimum thickness supported by a quasi-uniform mesh. This implies that the distance between two non-adjacent vertices cannot be smaller than $d_{thickness}$.

Definition 3.2. The maximum displacement step d_{move} allowed for a vertex is the value that prevents it to pass through a non-incident facet during the time evolution of the mesh.

Lemma 3.1. *Given a quasi-uniform mesh M characterized by d_{detail} and whose thickness does not exceed $d_{thickness}$, the maximum displacement step d_{move} allowed for one vertex satisfies the following inequality*:

$$4d_{move}^2 \leq d_{thickness}^2 - d_{detail}^2/3$$

Proof. The maximum value for the displacement d_{move} is found when a vertex V not incident to the largest possible face F of the mesh is initially located at the same distance $d_{thickness}$ from the three vertices of F and moves with the amount d_{move} perpendicular to F, while the vertices of F are moving with the same amount but in the opposite direction. The largest face on a d_{detail} tight mesh is obtained in the case of a regular triangle with edge length d_{detail}. From this we can deduce the inequality on d_{move} by applying the Pythagorean theorem to the triangle formed by V, the barycenter of F and one of its vertices (see Fig. 4). □

To avoid that an edge be reversed due to some tangential displacements of its vertices (which would make the normals of the adjacent faces point inwards), we also impose d_{move} to be shorter than half of the current minimum distance d_{min} between two adjacent vertices (see Fig. 5). d_{min} is a quantity that evolves dynamically during the mesh deformation, but its tendency is to stay close to the parameter d of the quasi-uniform mesh. In practice, choosing a constant value for d_{move} depending on d rather than d_{min} is sufficient.

Lemma 3.2. *Given a quasi-uniform mesh M characterized by d_{detail} and $d_{thickness}$, the embedding of M in a displacement field satisfying the previous constraints is guaranteed to be intersection and local inversion free.*

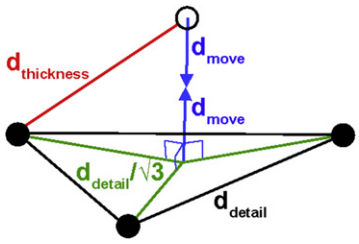

Fig. 4. d_{move} is chosen so that a vertex approaching a non-incident face that is moving in the opposite direction, is not allowed to pass through.

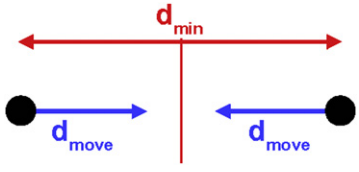

Fig. 5. In order to prevent local edge reversals, d_{move} needs to be smaller than the size of the smallest edge d_{min}.

After each displacement step, the compliance of the edges with d and the preservation of the d_{detail} tight property of the quasi-uniform mesh are performed as described in Section 2. Let us now describe our treatment of topological events.

3.2. Seamless splits and merges

The tight sampling of the evolving mesh and the sampling of the displacement field over time make it possible to detect an imminent self-intersection of the surface. That way, the changes of topological genus can be anticipated and be processed seamlessly, without any heavy computation of triangle intersection:

- Possible local overlaps of two non-neighboring parts of the surface are detected before they occur using simple collision-detection between spheres of diameter $d_{thickness}$ centered on the vertices by measuring the distance between points in the final position, instead of testing for more complex intersections between moving triangles as is would normally be done for a general deforming mesh.
- Changes in topological genus are triggered whenever two non-adjacent vertices get within a distance $d_{thickness}$ of each other. The merging between the neighborhoods of the two vertices is performed by connecting their 1-*rings* as illustrated in Fig. 6. The newly created edges that make this connection are denoted as *connecting edges*.

Once all such connections have been performed, the d_{detail} tight property of the mesh is restored as described in Section 2. Note that this recovery of the maximum length d_{detail} over the edges can result in the creation of new vertices on the connecting edges, which are not guaranteed to be at least at a $d_{thickness}$ distance from each other. The thickness property violation is temporarily relaxed for such vertices: they have been created to accompany the change in topological genus and if they are animated with movements that will separate them from each other during the next time step, then they will find themselves at a larger distance than $d_{thickness}$ when the protection is removed. We say that these vertices are *temporarily protected*.

Edge collapses and vertex join operations require an additional neighborhood cleanup that eliminates all thin parts such as pairs of triangles that have all vertices in common (Fig. 7) and locally separates the surface when two edges coincide (Fig. 8). Again the $d_{thickness}$ property is temporarily relaxed for the vertices that are separated in the process. The cleanup routine is the key to

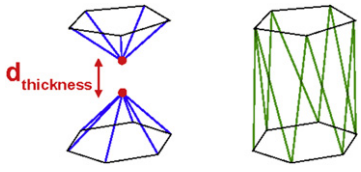

Fig. 6. Vertex join. When two non-adjacent vertices get within $d_{thickness}$ of each other, the surface changes topology by deleting them and reconnecting the remaining 1-*rings*.

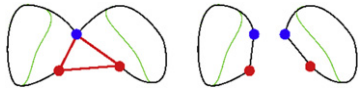

Fig. 7. Cleanup on thin parts. The two coinciding red triangles are erased and the blue vertex that is common to the two surface parts is split into two, which also divides the surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 8. Cleanup on thin parts. Each red vertex that is incident to the coinciding red edges is separated into two blue vertices, yielding a surface division in two separate parts. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

maintain a manifold mesh. This is discussed in more detail in the implementation (see Section 5.1).

4. Sculpting tools and user interaction

The method has been implemented in a simple sculpting application, featuring a series of tools that can be used both to deform the surface and to change its topological genus.

Our system accepts any type of displacement field and in particular all the fields already defined in professional applications, as well as some volume-preserving deformations described in previous research [21,1]. At each time step, the current user-defined field indicates a displacement to be applied to vertices inside a selected region. If the deformation exceeds the maximum allowed vertex displacement d_{move} then we first divide the initial deformation into the necessary number of elementary steps, as shown in Fig. 9 in the case of a common sweep deformation.

The deformation fields we use can be divided into two main categories:

- Fold-over free space deformations, defined independently from the mesh. They can range from simple sweeping fields to volume-preserving, radial or rotational fields. Among them, the fields we have implemented are: a simple deformation by sweeping (Fig. 9), the volume-preserving field defined in [21] (Fig. 10), a radial pinch-grow field that locally shrinks or expands the object and a trimming field that flattens the surface in the region where the tool is applied [14].
- Deformations depending on surface characteristics (normal, curvature, geodesic distance, Laplacian coordinates). These types of fields are more adapted, from an interaction point of view, to changes in topology since they are not fold-over free. We have implemented an inflate–deflate field depending on vertex normals and a smoothing operation that allows a user to perform Laplacian mesh editing [18].

Due to the specific nature of our mesh and its quasi-uniform sampling, changes in topological genus can occur each time two separate parts come closer than $d_{thickness}$ of each other, under any type of field, even the fold-over free ones, which do not normally allow overlaps.

4.1. Volume-preserving sweep deform

This type of deformation exemplifies a fold-over free displacement field that is applied by sweeping a spherical tool over a user-controlled path. The field is defined inside the spherical tool according to the volume-preserving deformations introduced by [21]. Fig. 10 depicts such a deformation in 2D to give a better idea of the kind of action obtained with this field.

The field is a piecewise defined differentiable function, with a constant value $\mathbf{v}$ inside an inner sphere of radius R_i, $\mathbf{0}$ outside the outer tool radius R_o and transitional in between. The general expression of the field is given by

$$\mathbf{D}(\mathbf{r}) = \nabla p(\mathbf{r}) \times \nabla q(\mathbf{r})$$

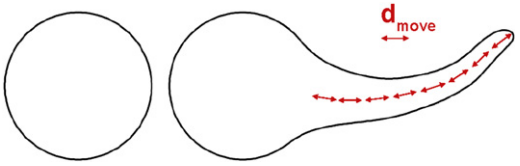

Fig. 9. Sweep deform. The original large deformation is divided into elementary steps (red segments) to limit the maximum displacement to d_{move}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

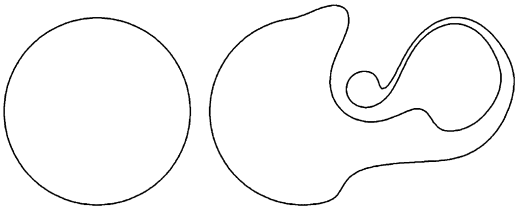

Fig. 10. Volume-preserving sweep deform.

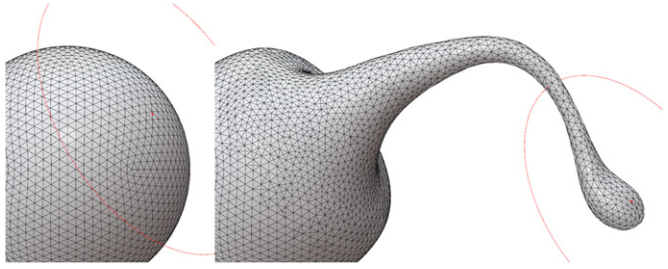

Fig. 11. Volume-preserving sweep deformation applied to a quasi-uniform spherical mesh.

where $\mathbf{r}$ is the position from the center of the tool and p, q are two scalar fields defined as follows:

$$p,q(\mathbf{r}) = \begin{cases} f,g(\mathbf{r}), & r \in [0,R_i] \\ f,g(\mathbf{r}) \cdot b((r-R_i)/(R_o-R_i)), & r \in (R_i,R_o] \\ 0, & r > R_o \end{cases}$$

with b a scalar blending function

$$b(x) = 3x^4 - 4x^3 + 1$$

f and g satisfying

$$f,g(\mathbf{r}) = \mathbf{u},\mathbf{w} \cdot \mathbf{r}$$

with

$$\mathbf{u} \times \mathbf{w} = \mathbf{v}$$

$\mathbf{v}$ being the constant field inside the inner tool region.

$\mathbf{u}$ is chosen arbitrarily in the plane perpendicular to $\mathbf{v}$, and $\mathbf{w}$ is added so that the three vectors form an orthonormal basis. As long as the base is orthonormal the field will be symmetric with respect to the direction of $\mathbf{v}$.

The resulting field is divergence-free and therefore volume-preserving as proven by Theisel et al. [20] and $C(1)$ continuous which insures smooth deformations. However, we do not compute the path integration used in the original paper by von Funck [21], because our quasi-uniform mesh does not allow details to get smaller than d_{detail}. Dividing the path into elementary steps of size d_{move} and applying the field discretely is sufficient to capture a good d_{detail} approximation of the final shape (Fig. 11).

4.2. Inflate–deflate

Inflation is an operation that only depends on the normal at each vertex and therefore is not a fold-over free deformation. For each vertex the displacement is aligned with its normal and the direction is either positive for inflation or negative for deflation as shown in Fig. 12.

The expression of the field is given by

$$\mathbf{D}(\mathbf{r}_i) = \varepsilon \cdot b(r_i/R) \cdot \mathbf{N}_i$$

where $\mathbf{r}_i$ is the position of the vertex i with respect to the center of the spherical tool, $\mathbf{N}_i$ is the normal at vertex i, b is a decreasing function defined inside the interval $[0..1]$ with values in the same interval and R is the radius of the tool. ε can take the values ± 1 for inflation and deflation respectively. We used a simple polynomial expression for the blending function b of the form

$$b(x) = (n-1)x^n - nx^{n-1} + 1$$

with the integer parameter $n > 2$ that insures the first derivative to be 0 for $x=0$ and 1 and gives a smooth transition between the deformed and non-deformed regions while also allowing some control over the extent on which the deformed region is modified (Fig. 13). However, our choice of the function b remains purely arbitrary and more complex expressions could be easily handled according to the needs.

Fig. 14 exemplifies a change in topological genus produced by an inflation.

4.3. Twist

Twist is an operation that performs a rotation centered on the point of application of the tool, around the surface normal in that point, with an amplitude that decreases with the radius inside the tool (see Fig. 15).

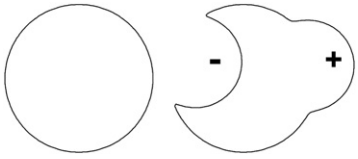

Fig. 12. Inflate (+) and deflate (−) applied to a sphere.

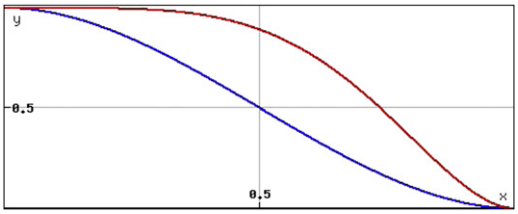

Fig. 13. Blend function for $n=3$ (blue) and $n=6$ (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

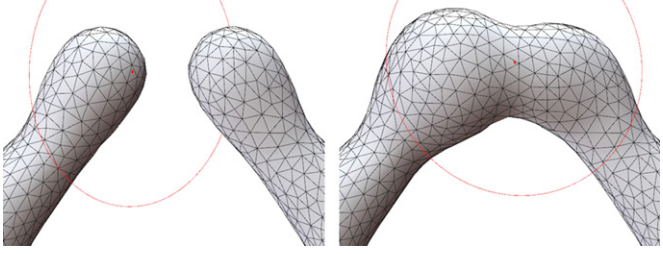

Fig. 14. Change in topology caused by an inflation.

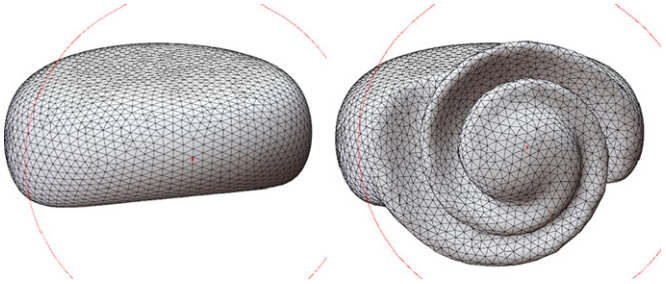

Fig. 15. Twist operation.

The expression of the field inside the spherical tool is given by

$$\mathbf{D}(\mathbf{r}) = (\mathbf{r} - (\mathbf{r}\cdot\mathbf{N})\mathbf{N})(1-\cos A(r)) + \mathbf{N}\times\mathbf{r}\sin A(r)$$

where $\mathbf{r}$ is the vector from the center of the tool, $\mathbf{N}$ is the surface normal in the point where the tool is applied and also the vector around which the rotation is performed and $A(r)$ is the angle of rotation that decreases with the radius r as described in the formula:

$$A(r) = A_0 \cdot b(r)$$

where A_0 is a constant and b is the same blending function as in the previous deformations:

$$b(x) = (n-1)x^n - nx^{n-1} + 1$$

with $n > 2$.

5. Results and discussion

5.1. Implementation

The implementation of the method is straightforward: a series of operations are performed on an input manifold mesh in order to first create and then to maintain a quasi-uniform sampling and preserve the manifold property. The mesh operators – edge collapse, edge split, vertex join – act when the properties described in Sections 2 and 3 are violated. To maintain a manifold mesh, a cleanup operation is performed locally on vertices affected by mesh operators.

Mesh structure: We describe the mesh using a compact array-based data structure, with each facet storing the indices of adjacent facets and incident vertices in the corresponding arrays and with each vertex storing the index of one incident facet. When mesh operators cause the removal of certain components, the structure updates in a lazy manner: the deleted components are marked as inactive during the deformation and are removed only at the end of each user action or after a large number of steps. This way the removal of components does not significantly slow down the process.

Manifold mesh: In order to maintain the manifold property a local cleanup is performed in the neighborhood of each vertex that is modified by a mesh operator. Cleanups are performed on the vertex obtained by collapsing an edge and on all the vertices of connected 1-*rings* after a vertex join operation. The procedure eliminates all degenerate components (represented in green in Fig. 16) in the neighborhood of the affected vertex. This happens as follows:

- If two consecutive triangles in the neighborhood are degenerate (Fig. 16a) – i.e. share the same vertices – they are deleted and the connectivity among the remaining components is reconstructed.
- If two non-consecutive triangles are degenerate (Fig. 16b), they are deleted and the central vertex and the neighborhood

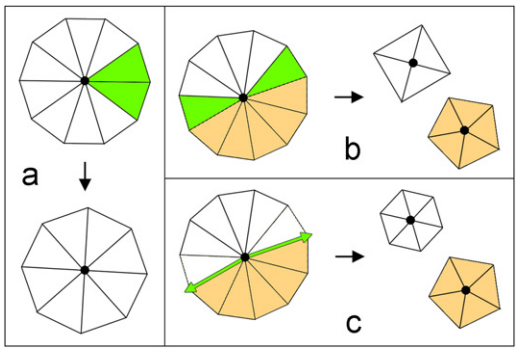

Fig. 16. Cleanup on the triangle fan around the affected vertex (in black). Degenerate components are schematically shown in green and should be considered collapsed over each other. For simplicity they are represented as separate components in the neighborhood. The procedure is applied as follows: (a) Degenerate consecutive facets in the fan are deleted. (b) Degenerate non-consecutive facets are deleted and the central vertex is split. (c) Degenerate edges split the neighborhood into two regions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is split into two separate regions. We continue recursively from the first step on the two separate neighborhoods.

- Degenerate edges (Fig. 16c) split the central vertex and the neighborhood into two separate regions. Again we continue recursively from step one on the two separate parts.

Changes in topology: Topology changes are achieved through vertex join operations, which consist of deleting non-neighboring vertices that come in close proximity of each other and welding the remaining parts.

Connecting 1-*rings* is currently implemented in a simple manner by using the Bresenham line algorithm [5]. This is normally used to rasterize segments on a screen, but instead of marching X points horizontally and at the same time going Y points vertically, we use it to connect the X vertices on the first 1-*ring* with the Y vertices of the second 1-*ring*. This implementation can at times lead to self-intersecting triangles, but this issue can be solved by using more robust methods like the reconstruction from unorganized cross-sections by Boissonnat et al. [4]. In practice though, the problem can always be eliminated through local smoothing.

If the two 1-*rings* share common vertices, we can still use the previous reasoning, but this time we apply the procedure on each pair of corresponding disconnected parts of the two rings.

We distinguish the special case when the two 1-*rings* of joining vertices share only an edge. Here we simply test if a flip of the common edge, that would connect the two joining vertices, would also violate the compliance property. If true the edge is flipped and then collapsed. This helps to eliminate some rare cases in which the non-compliance with d might result in a distance smaller than $d_{thickness}$ between two second order neighbors and prevents the creation of unwanted topology in the form of local handles.

The vertex join operations are performed sequentially, so two approaching flat surfaces will first unite through small bridges before completely merging. In practice this works well and has no unwanted influence on future deformations, but it might create visually poor outcome if the deformation is stopped in an intermediary phase.

Choice of parameters: Once the level of detail – d_{detail} – is fixed, the other important parameters d, $d_{thickness}$ and d_{move} can be chosen with a certain flexibility without the model breaking up.

Formally d_{move} depends on the size of the smallest edge in the deforming region. Even if theoretically we cannot guarantee a lower bound for d_{min} our method maintains its value close enough to d so that we do not see any problems during deformation. We can even choose a constant value depending on d without encountering any inversions on the surface. However, this dependence on d_{min} remains an open issue.

The value of $d_{thickness}$ is chosen close to the lower bound set in Lemma 3.1, in order to prevent the creation of unwanted topology between distant parts. An upper bound is hard to define since the compliance with d cannot be guaranteed.

5.2. Results

The quasi-uniform mesh with self-adaptive topology is very intuitive and easy to use for sculpting purposes, as illustrated in the attached examples (Figs. 17 and 18 and Teaser image), which are all direct captures from our application during the sculpting process.

In each figure we give the value of d_{detail} to show the approximate size of a detail relative to a 1 m sculpture of the object. It should be noted that this is different than having a grid with d_{detail} precision since vertices can be positioned in arbitrary positions in space and triangles can have any orientations instead of those obtained from the reconstruction in a grid-based method.

Although we have only implemented a few simple tools, we can already create detailed shapes, with complex topology. All the models were created by an artist accustomed to professional software, running our interactive sculpting application on a laptop with Intel i5 processor and nVidia GeForce 310M graphics card. Mesh sizes range from 50k points to 150k points and the time taken to sculpt them varies from 1 to 3 h.

We have also tested the performance of several large scale deformations, while changing some characteristics of the surface. One important parameter that describes the quasi-uniform mesh

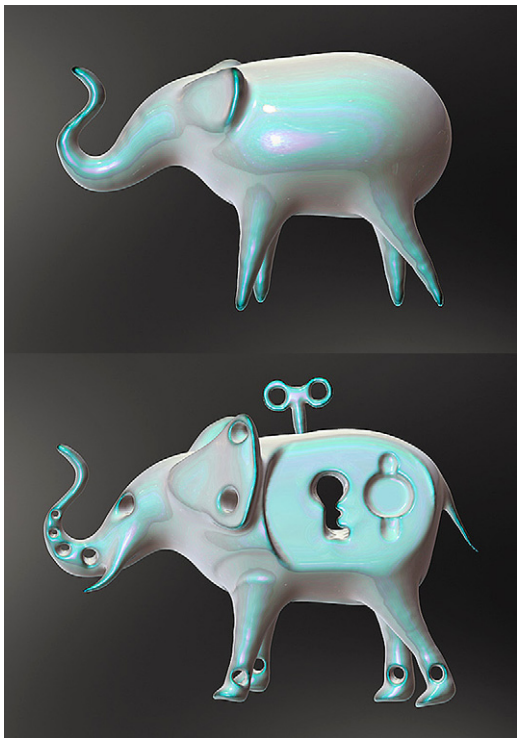

Fig. 17. Example sculpture created from a sphere by a professional artist. Changes in topological genus can be seen from one frame to the next. Time taken: 1–2 h. Mesh size: 80k points. d_{detail} for a 1 m statue of the object: 6 mm.

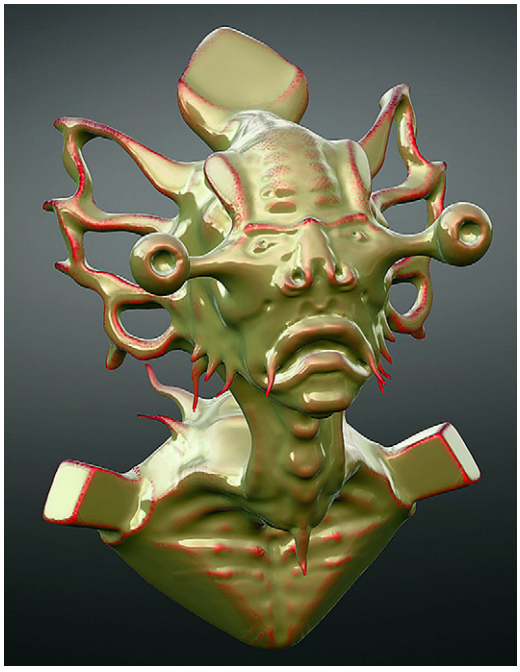

Fig. 18. Example sculpture created from a sphere by an artist using four tools: sweep deform, inflate–deflate, smooth and trim (for creating flat surfaces). Time taken: 2–3 h. Mesh size: 150k points. Rendering is done with simple shaders as a function of local curvature. This is a real-time capture from the application. d_{detail} for a 1 m statue of the object : 4 mm.

Table 1
Performances of our dynamic surface sculpting framework in terms of vertex creation and deletion as a function of $r=d_{detail}/d$. The variation in FPS between deformations should not be taken into account since each of them is different in nature. The lowest FPS are obtained for the sweep deformations because they give the largest displacements among the cases we studied. The FPS and total time values show that for usual deformations the visual feedback is real-time.

Deformation	# created vertices	# deleted vertices	Time (ms)	FPS avg-min
7292 initial vertices, $r=d_{detail}/d=2$				
Sweep	24215	18327	2268	14-12
Sweep V=ct.	27386	23166	2150	15-13
Inflate	23397	15436	1510	184-159
Deflate	11493	9506	1410	235-205
Grow	21360	16447	2430	39-36
Trim	622	806	2770	51-47
7292 initial vertices, $r=d_{detail}/d=2.05$				
Sweep	19526	13398	2120	15-13
Sweep V=ct.	21821	17395	2010	16-13
Inflate	19578	11239	1410	186-160
Deflate	9407	7333	1370	238-207
Grow	14668	9522	2180	43-41
Trim	422	582	2530	56-53
7292 initial vertices, $r=d_{detail}/d=2.2$				
Sweep	13934	7187	2246	14-12
Sweep V=ct.	15917	10912	2140	15-13
Inflate	15029	6069	1420	173-147
Deflate	7370	5024	1390	231-201
Grow	9305	3437	2400	38-35
Trim	95	231	2760	51-48

is the ratio $r=d_{detail}/d$. The deformations were applied on a sphere and the statistics on the number of created and deleted vertices for the basic deformation tools used in our framework are reported in Table 1.

The important result is observing the relation between the number of created and deleted vertices and the time taken to perform the deformations, as r changes.

We can see that there are two opposing effects. On one hand, as r increases, d decreases which in turn determines predominant smaller edges and therefore the maximum vertex displacement d_{move} will take a smaller value and more steps will be needed for a large deformation. This results in more time spent applying the displacement field. On the other hand, as r decreases and the value of d_{detail} and d become closer, there is an increase in the number of both the created and destroyed vertices which means more time spent on edge split and collapse operations. As the total time for a large deformation is

$$T_{total}=(n_C\cdot T_{split}+n_D\cdot T_{collapse})+(N_{steps}\cdot T_{deform})$$

we see that we can find an optimal value for r, depending on the deformation and namely on the time T_{deform} needed to calculate the displacement field for each vertex as well as on the small variation in number of vertices. In practice we use a single value $r=2.05$ based on the tests shown in the table, and which is reasonable for all the deformations we have implemented.

5.3. Advantages and limitations

The quasi-uniform mesh structure we developed for sculpting has one major advantage over both adaptive and multi-resolution polygonal methods: its ability to easily handle any change in topological genus. Another important feature is the unambiguous tracking of any deformation by dividing it down to the level of detail allowed by the structure.

Obviously, our model is not well adapted for the creation of very large objects with some local tiny details: here, adaptive sampling approaches would be more efficient. Another issue is the difficulty in maintaining sharp features during deformation due to the mesh operators that are applied in the deforming regions. Since our mesh adapts its topology over time, parallelizing it on the GPU is also more difficult than with multi-resolution models, so our model is less efficient in terms of speed and mesh size than [14], which processes 10M vertices at interactive rates but with the drawback of imposing a fixed connectivity and topology.

Note that global refinement of sampling resolution is easily done in our framework, without losing the adaptive topology feature: the user could start sculpting with a coarse quasi-uniform mesh, and refine it uniformly to add finer details when the basic shape is set.

If we compare our model with grid-based methods, which also seamlessly handle topological changes, the main advantage of our structure is the temporal coherence of mesh connectivity, which is only locally modified from time to time: this should facilitate the addition of surface texture or bump mapping in future work. In contrast, methods that sample polygons on a grid, such as the one used in 3D Coat [13] enable both deformations and changes in topological genus, but lose temporal coherence of the mesh and also suffer a loss in triangle quality when the shape moves, because of the frequent re-samplings. Moreover, our choice to use a time-coherent triangular mesh enables some acceleration by storing part of the data on VBOs (vertex buffer objects) until it is actually modified and using it only for fast rendering in the meantime. Meshes can also be interactively deformed with a large variety of fields without any complex computation.

At each step our data structure is automatically refined where needed and simple collision tests between spheres are carried out to prevent intersections. These tests are a lot simpler than testing for self-intersections on an adaptive mesh. Although this last operation facilitates the handling of changes in topological genus,

it remains the most time consuming feature of our method. Collisions must be calculated between spheres of diameter $d_{thickness}$ centered on the mesh vertices. If we calculate this for each pair of vertices in the selected region we have a complexity of $O(n^2_{selected})$ with $n_{selected}$ the number of selected vertices. We improved this by using spatial subdivision where the space is partitioned into a regular grid and collisions are calculated only between spheres in adjacent cells. This could be further improved using an implementation on the GPU, which would perform collisions 20–30 times faster than the spatial subdivision algorithm on the CPU as shown in GPU Gems 3 [17] and overall gain an order or two in efficiency over the current implementation.

As it is – without any of the hardware accelerations we mentioned (region VBOs or GPU collisions) – our sculpting system is already real-time for all of the examples of complex topology and reasonably high detail presented in this paper.

6. Conclusion and future work

We have presented the first mesh-based sculpting system that seamlessly handles arbitrary changes in topological genus, thanks to a simple, quasi-uniform mesh structure. Our system does not require the user to know anything of the underlying representation, besides the fact that the material acts at a given level of detail. He interacts with gestures, only concentrating on the shape he is creating. Similarly to grid-based virtual-clay approaches, automatic merges and splits happen at a given resolution, which is very intuitive since it is close to the behavior of real clay.

In future work we plan to extend our framework to arbitrary non-manifold meshes. We also intend to handle fast approximate and robust Boolean operations between two quasi-uniform meshes by taking advantage of the tight sampling. Another important aspect will be the addition of surface color or texture during sculpting which can bring even more detail to the surface without much increase in memory. We would also like to focus on the problem of reconciling our quasi-uniform mesh with an adaptive sampling which would allow a user to easily sculpt objects with variable level of detail, while also taking advantage of changes in topological genus. We also have an idea on how to sculpt objects with sharp-features by protecting corresponding edges against certain mesh operators. All these new mechanisms could make interesting additions to our sculpting system while still keeping a reasonably simple underlying structure and an intuitive user interaction.

Acknowledgments

We would like to thank the anonymous reviewers for their extensive comments and suggestions that helped to improve our paper. This work has been supported and funded by the ANR Triangles, contract number ANR-07-BLAN-0319, and by the LIMA project of the Rhône-Alpes Region research Cluster ISLE.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version of 10.1016/j.cag.2011.03.033.

References

[1] Angelidis A, Cani M-P, Wyvill G, King S. Swirling-sweepers: constant volume modeling. In: 12th Pacific conference on computer graphics and applications, Pacific graphics 2004, October, 2004, Seoul, South Korea. p. 10–5.
[2] Autodesk, 2010. Mudbox. URL ⟨usa.autodesk.com⟩.
[3] Blender Foundation, 2010. Blender. URL ⟨www.blender.com⟩.
[4] Boissonnat J-D, Memari P. Shape reconstruction from unorganized cross-sections. In: Proceedings of the fifth Eurographics symposium on geometry processing. Aire-la-Ville, Switzerland: Eurographics Association; 2007. p. 89–98. URL ⟨portal.acm.org/citation.cfm?id=1282003⟩.
[5] Bresenham JE. Algorithm for computer control of a digital plotter. IBM Systems Journal 1965;4(1):25–30.
[6] Debard J-B, Balp R, Chaine R. Dynamic Delaunay tetrahedralisation of a deforming surface. The Visual Computer 2007:August;12. URL ⟨liris.cnrs.fr/publis/?id=3164⟩.
[7] Dewaele G, Cani M-P. Interactive global and local deformations for virtual clay. Graphical Models 2004;66(6):352–69. [Special issue: Pacific Graphics 2003].
[8] Ferley E, Cani M-P, Gascuel J-D. Virtual sculpture. In: Eurographics '99, September 1999 [Short paper].
[9] Ferley E, Cani M-P, Gascuel J-D. Resolution adaptive volume sculpting. Graphical Models 2001;63(6):459–78. [Special issue on Volume Modelling].
[10] Galyean TA, Hughes JF. Sculpting: an interactive volumetric modeling technique. In: Computer graphics (Proceedings of SIGGRAPH 91), vol. 25; 1991. p. 267–74.
[11] Lachaud J-O, Montanvert A. Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction. Medical Image Analysis 1999;3(2):187–207.
[12] Pettersson T. 2010. Sculptris. URL ⟨www.sculptris.com⟩.
[13] Pilgway, 2010. 3D-Coat. URL ⟨www.3d-coat.com⟩.
[14] Pixologic, 2010. ZBrush. URL ⟨www.pixologic.com⟩.
[15] Pons J-P, Boissonnat J-D. A Lagrangian approach to dynamic interfaces through kinetic triangulation of the ambient space. Computer Graphics Forum 2007;26(2):227–39.
[16] Schmitt B, Pasko A, Schlick C. Constructive sculpting of heterogeneous volumetric objects using trivariate b-splines. The Visual Computer 2004;20(2). URL ⟨www.labri.fr/publications/is/2004/SPS04⟩.
[17] Scott Le Grand, NVIDIA Corporation, 2010. GPU Gems 3. URL ⟨developer.nvidia.com/object/gpu-gems-3.html⟩ [Chapter 32].
[18] Sorkine O, Cohen-Or D, Lipman Y, Alexa M, Rössl C, Seidel H-P. Laplacian surface editing. In: Proceedings of the Eurographics/ACM SIGGRAPH symposium on geometry processing. Eurographics Association; 2004. p. 179–88.
[19] Sugihara M, Wyvill B, Schmidt R. WarpCurves: a tool for explicit manipulation of implicit surfaces. Computers & Graphics 2010;34(3):282–91. [Shape Modeling International (SMI) 2010].
[20] Theisel H, Weinkauf T, Hege H-C, Seidel H-P. Topological methods for 2D time-dependent vector fields based on stream lines and path lines. IEEE Transactions on Visualization and Computer Graphics 2005;11(4):383–94.
[21] von Funck W, Theisel H, Seidel H-P. Vector field based shape deformations. In: ACM SIGGRAPH 2006 papers. SIGGRAPH '06. New York, NY, USA: ACM; 2006. p. 1118–25. doi: acm.org/10.1145/1179352.1142002.