

Adaptive Hierarchical Mesh Detail Mapping and Deformation

Bruno José Dembogurski[1(✉)] and Anselmo Antunes Montenegro[2]

[1] Universidade Federal Rural do Rio de Janeiro, Rio de Janeiro, Brazil
brunodembogurski@ufrrj.br
[2] Universidade Federal Fluminense, Rio de Janeiro, Brazil
anselmo@ice.ufjf.br

Abstract. In this work, we present a new method for controlled deformation and detail addition to 3d shapes represented as variable resolution meshes. The input data is a surface with arbitrary genus, represented by a polygonal mesh, and a set of parameters for edition control: positional information, the level of resolution, mesh features and direction of propagation of the deformation. An adaptive hierarchical mesh structure is constructed using an iterative feature-sensitive simplification method that concomitantly generates the parameterization of the mesh. The coarsest level of the representation defines the base domain which stores the original geometry via a local parameterization process. We apply local modifications to the base domain according to predefined functions; a noise function for details or any geometric deformation. In the sequel, the deformation of the base mesh is propagated to the original mesh. Our main contribution is a method that relies on the power of adaptive hierarchical structures to generate details with a greater degree of control by using a set of operators that explore the data structure properties as well as the information extracted and computed from the mesh.

Keywords: Detail mapping on meshes · Mesh editing · Hierarchical meshes · Feature-sensitive decimation · Mesh processing

1 Introduction

An important problem in the area of geometry processing is mesh geometry editing, which can be burdensome when a large amount of vertices must be manipulated to make a meaningful change [7]. To deal with such a problem, it is usually necessary to build new representations on top of the raw input data. This representation can be obtained, for example, by fitting spline models or computing sophisticated computational manifold structures. Frequently, a parameterization processes has to be computed in order to construct an appropriate representation, what might become a complex task in the case of surfaces with genuses that are different from zero.

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 273–288, 2019.
https://doi.org/10.1007/978-3-030-24289-3_21

In this work, we investigate the problem of detail addition and mesh deformation with special emphasis in controlling the mapping of procedurally generated details onto shapes described by meshes. There are many works in the literature that tackle issues related to mesh editing. Nevertheless, as far as we are concerned, very few works deal with the problem of controlling procedurally generated details using geometrical and topological information to achieve a better control of the procedural detail mapping process. We propose the use of a variable resolution parameterized mesh representation and a set of deformation operators that take for granted the adjacency information, hierarchical structure and adaptability of the mesh representation to guide the deformation. We demonstrate by experiments that the proposed representation and the designed operators can offer sufficient level of control and expressiveness power to produce many different deformations and detail addition effects. An example of our method steps can be seen in Fig. 4.

The main contributions of this work are:

- A new method for deforming and adding details to meshes with arbitrary genus in an adaptive and controlled way, based on a combination of variable resolution mesh representations, operators for procedural noise generation and mesh subdivision.
- The proposal of a *general deformation handle* that generalizes the idea of *deformation handle* [2] in mesh editing. General deformation handles consider mesh features, the level of resolution and direction of propagation of the deformation.
- An improvement in the process of base domain generation for variable resolution mesh representations by using feature-preserving decimation. This makes features available for controlled edition even in the coarsest levels of the mesh representation differently from multiresolution mesh editing based on fairing.

2 Related Work

According to [4], it is possible to group mesh deformation techniques into two categories: *surface-based deformations* and *space-based deformations*. Surface-based deformations offer a higher degree of control as vertices can be manipulated and constrained individually whereas space-based deformations define deformation implicitly by deforming space. Deformation methods can also be classified as *linear* or *non-linear* depending on the type of energy minimization required for computing the solution based on optimization strategies. Our work can be classified as a topological variable resolution surface-based deformation controlled by the mesh vertices' position, features detected, the levels of resolution of its vertices and the direction of deformation. This section presents a brief description of works related to surface-based multiresolution and feature-aware approaches. For a more general review of shape editing and deformation we refer the reader to [4,17].

Multiresolution Deformation. Multiresolution deformation techniques decompose the original surface into a low-pass filtered coarse approximation and high-frequency details. Modifications are applied to the coarse version of the surface, e.g., by *transformation propagation* [2] or any arbitrary editing technique. Finally, the stored high-frequency details are added to the edited version of the coarse approximation. In the literature we can find two main approaches for building multiresolution mesh representations: *subdivision-based* [3,21,24] and *simplification-based* [9]. In [24], Zorin et al. propose a multiresolution representation that is constructed using a combination of the Loop subdivision scheme with Taubin's faring technique [18] in which details are encoded in local frames. Velho et al. in [21] also propose a similar multiresolution representation but differs by using the Catmull-Clark subdivision scheme. Smoothing is done through a quasi-interpolation and the mesh is represented as an atlas. One important characteristic of [21] is that they also deal with the problem of procedural detail creation. Subdivision-based representations require meshes with subdivision connectivity and also have the drawback that the features that are supposed to be edited do not necessarily match the topology in the coarser levels.

Progressive Mesh Editing. In [6], Derzapf et al. present a method for parallel editing of *Progressive Meshes* [8], a data structure for representing meshes as a coarse mesh associated with a set of edge collapse operations and their local and global attributes. Derzapf et al. propose a CUDA GPU implementation of a new simplification algorithm that enables edition of progressive meshes in real time. In their method, they propose the storage of all operations based on the labeling of vertices after computing edge collapses in their data structure. Modifications are automatically propagated to finer resolutions using an encoding of the split operations based on local coordinate systems. The way they propagate modifications is similar to ours since the geometrical information of each vertex is also stored in a local coordinate system. The main difference is the usage of the *edge-weld* operation instead of *vertex split* and we do not store any operation. Furthermore, differently from our method, the approach presented by Derzapf et al. cannot change the connectivity of the mesh.

Adaptive Multiresolution Meshes. Maximo et al. [13] present an adaptive multiresolution mesh representation exploring the computational differences between the CPU and the GPU. They consider a dense mesh as input and simplify it to a base domain in a similar fashion to the work presented in [11], however, using an atlas structure. Its main objective is to show the adaptive control of the mesh resolution in CPU-GPU coupled applications. Differently from Maximo, we use an A4-k mesh representation instead of an A4-8 because the feature sensitive decimation used in our representation construction produces meshes with vertices that have valency different from four and eight.

Semantic Mesh Editing. In [23], Yumer et al. deal with the problem of controlled mesh editing in a very different and innovative way. They propose a method for creating deformations based on semantic attributes rather than using detailed geometric manipulations. For this, they formulate deformations as a problem of

constrained path traversal in a geometric space of shapes where models in the shape set are seen as regression points. Thus, a database of shapes can be seen as a continuous deformation space. They argue and show via use cases that a non-expert user can produce semantically guided shape variations by using their exploration and design interface in a way that would be quite difficult using other approaches.

Feature-Aware Shape Edition. Dekkers et al. proposed in [5] a new mesh editing scheme that supports both elastic deformation, via a modification of a conventional Laplacian editing method, and plastic deformation by adding and removing triangles in low salience regions that absorb most of the deformation energy.

3 Problem Definition

Given a mesh M and a *general deformation handler GDH* represented by a set of 3-ples (R, L, VF) where $R \subset M$ defines a region of interest, L represents the levels of detail of the vertices in R, and VF is a vector field, compute a new mesh M' by defining as transform $T : M \rightarrow M'$ where T is a composition of deformation operators D_i.

4 Methodology

Mesh edition based on multiresolution analysis and synthesis requires mesh fairing which eventually removes important features that may be used for controlled edition. In this work, we propose a different approach that tries to overcome this problem. It is based on the representation of the surface as a parameterized variable resolution mesh coupled with a set of extensible deformation operators. Our reasoning behind this choice takes in consideration that variable resolution meshes are able to express adapted mesh hierarchies with an overall better preservation of important features, making it possible to achieve a more precise control of the deformation even when applied at coarse resolutions. Since the representation is a completely parameterized adaptive hierarchical topological structure, it is possible to navigate through the mesh's levels of resolution and spatial adjacency to perform any sort of geometry deformation and detail addition, which can be smoothly propagated across the regions of the mesh. The result of such process can be propagated to the most refined levels of the shape representation in a simple way.

5 Proposed Method

Our method is composed of three main steps: (a) representation construction (by concomitant decimation and parameterization), (b) refinement and (c) deformation/detail generation which are described below.

5.1 Construction of the Mesh's Representation

Our mesh representation is based on the variable resolution a4-k structure introduced by Velho in [20]. This is a powerful structure for the representation of objects at multiple levels of detail. The hierarchical structure of the variable resolution a4-k mesh is built using a simplification process that uses a set of local modifications defined on clusters of two triangle faces. These modifications are made through the *Stellar Operators edge flip*, *edge split* and *edge weld* and cause minimum changes in a local neighborhood. The refinement and deformation/detail addition can be performed at any level or region of the mesh. As new vertices are inserted or modified, they are reparameterized producing a very powerful representation.

Mesh Simplification. Our mesh simplification algorithm is a feature-sensitive extension of the *Four-face Cluster Technique* presented in [19].

Feature lines are one of the most prominent characteristics of a surface where sharp details usually appear. These are extremely important to this work, because the preservation of important features in the hierarchy, especially in the base mesh, allows a greater level of positional and directional control in the deformation and detail generation process. This is even more evident for meshes representing man-made artifacts. In our approach, we use the *Tensor-feature Analysis* as proposed by [14] couple with a *Voting Scheme* and the *Neighboring Vertex Coincidence* (NVC) criterion, introduced by Wang in [22], that separates weak-edge vertices from noise ones.

In order to guarantee that the main features are preserved in the base domain mesh through the simplification procedure, we mark all features we want to be preserved as *unremovable*. After this process, an independent set of clusters that covers most of the mesh is selected. During this step, every vertex that is related to an important feature of the surface is marked as unremovable and is not included in the cluster list, thus maintaining feature lines and corners untouched. The cluster simplification creates a geometric modification in the 1-ring neighborhood of faces of the vertex being removed, meaning that both boundary vertices and edges remain unchanged.

The feature preservation scheme creates a less uniform distribution of valencies across the mesh. This is one of the reasons for using an a4-k mesh since the maintenance of an a4-8 structure under this circumstance would be a hard task, requiring frequent reparameterizations.

Mesh Parameterization Guided by Simplification. Our method uses a parameterization approach based on the MAPS [11] parameterization. Although other global parameterization methods could be used, the MAPS scheme is compatible with our idea of concomitant parameterization and hierarchical construction, and is able to deal with arbitrary genus surface without the need for cutting the mesh to generate a structure with the topology of a disk. Moreover, it is very simple to include our feature-preservation scheme in the MAPS-based paremetertization.

After the simplification process, the resulting structure is a hierarchical parameterized mesh with initial levels $M^0, M^1, M^2, \ldots M^{l_{max}}$, where l_{max} is the level of the coarsest mesh. As the deformation process is performed, such mesh levels of resolution can be modified in a adaptive way relying on the nice properties of the a4-k mesh structure.

5.2 Deformation

In our approach deformation operators are used to deform the mesh. A discrete deformation operator is defined as a map $D : M \times (R, L, VF) \rightarrow M'$ where R is a region of interest defined by a subset of the vertices V of the mesh together with its adjacency relationships, L is the level of detail of the vertices in R and VF is a vector field defined on R, that guides the direction of displacement.

One of the fundamental properties of the proposed operators is their capability to use the level of detail of the vertices in a given region to modulate the intensity of the deformation. It also may take vector fields to define directional constraints on the deformation enabling greater control as well as the adjacency information of the vertices $v \in R$.

Parameterization Update After Deformation. After applying any combination of deformation operators to the mesh, we must update the coordinates of the vertices or define the parameterization of newly inserted vertices. This update is performed once, immediately after the completion of the detail addition process. First we perform a simple check, verifying whether a vertex v in a given triangle is supposed to be deformed or not. The coordinates of a deformed vertex v are updated using the barycentric coordinates $\alpha(v), \beta(v), \gamma(v)$, defined by the parametrization, in conjunction with its detail represented by its height value $h(v)$ in the local frame. The height value $h(v)$ is also used to retrieve the 3D coordinates of a vertex in the mesh's global refinement step which reconstructs the finest levels of representation of the mesh from the base mesh.

5.3 Adaptive Refinement

In order to reconstruct a mesh level K^{l-1} from the current mesh in level K^l we must walk across the triangles of K^l checking which vertices in the parametrized structure needs to be re-inserted. Each triangle is associated to a multimap data structure containing its parameterized vertices grouped by their levels of detail defined just before they are removed in the simplification step. Each group of vertices with the same level l can be retrieved using the access key given by l. Thus, to obtain the vertices that must be re-inserted at each level l, we only need to recover the entries with that key value. These vertices are then stored in a temporary queue ordered by their removal order. This guarantees that the insertion order respects the order of vertex removal, by *edge weld*, in the simplification process.

An *edge split* operation is used to re-insert a vertex. To decide which edge must be split when a vertex is inserted, we compute its Euclidean distance to the triangles candidate edges and select the one that is closer.

During parameterization, the vertices of a given triangle may be parameterized onto different triangles on the base domain [11]. These singular cases generate overlapping edges during the refinement process and must be dealt with to properly rebuild the original geometry. A common approach is to subdivide such triangles until all vertices that belong to the same triangle are re-parameterized to the same triangle in the base domain [13].

In our approach, we used a heuristic solution in which we do not fix overlaps in the parameterization, doing so in the insertion operation. During the insertion step, we check if the distance from the current vertex position p to the center of the edge e to be split is greater than the length of any of the edges $(e_{t_1}, e_{t_2}, e_{t_3})$ of the new triangle t. If this is true, we search for the right insertion edge e_{n_t}, which, usually, is in one of the neighboring triangles n_t.

There are also challenges regarding the reconstruction of the original surface that must be tackled. A particularly tough one is how to restore the original mesh state when operations of vertex valence correction (edge flips) were needed. Intuitively, while refining, we want the vertex degree to be the same as before it was when removed. To achieve that, a vertex degree check must be performed, verifying if the inserted vertex has its original degree. If this is not true, we must perform *edge flip* operations to correct the vertex degree. After correcting a vertex degree in the mesh refinement step, we approximate the geometry of the original input mesh. The experiments have shown that only very few triangles have different geometry when compared with the original mesh and their aspect ratios are similar to the original ones.

6 Operators

We present a set of operators that are used to compute the deformation of the mesh geared by the regions of interest, the level of detail of its elements and a directional vector field. These operators can be divided into three groups: *auxiliary operators*, *deformation operators* and *composite operators*. Auxiliary operators do not perform deformations on the geometry of the mesh but carry on actions that aid the deformation process. There are two auxiliary operators: one responsible for a local refinement of the mesh, the *subdivision operator*, and a second one, the *feature-based query operator* which is used to select regions of interest satisfying some criterion. The deformation operators are used to modify the geometry of the mesh. We defined here three basic deformation operators: the *smoothing operator*, the *procedural detail operator* and the *geometric operator*. *Composite operators* combine two or more auxiliary and/or deformation operators to compute more complex effects.

6.1 Subdivision

The proposed subdivision operator explores the local adaptivity properties of the a4-k structure to deal with cases where the region of interest in not sufficiently refined for the creation of a specific deformation effect. It generates new samples by creating a concentric subdivision pattern around a valency four vertex from its link towards the center of the subdivided region where the vertex lies in. Figure 1 shows that this process only modifies the region restricted to the original 1-ring of the selected vertex. The use of such operator in a specific region of interest R requires that the vertices $v \in R$ define an independent vertex set. We denote the subdivision operator by *sub* (R) where R is a region of interest composed of a set of independent vertices. Whenever a vertex has valency different from four it must have its connectivity changed by using the *edge flip* operator so that subdivision can be performed.

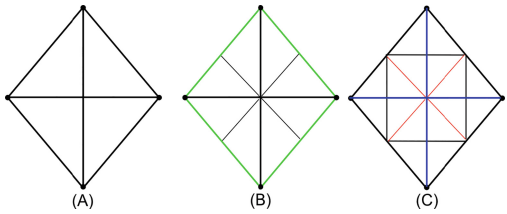

Fig. 1. (a) Input example of a degree four neighborhood. (b) One step of subdivision, where all link edges (green) are split. (c) Second step, where all even edges (red) are split. Every step following this one, switches between even (red) and odd edges (blue) for splitting. (Color figure online)

6.2 Smoothing

When combined with other operators the subdivision is usually followed by the use of a smoothing mask for smoother results. We based our smoothing operator on the smoothing component of the Loop subdivision scheme presented in [12]. The smoothing operator is represented by *smt* (v) and takes as an input a vertex v. We decided to separate the smoothing operator from the subdivision as their use in combination is not mandatory.

6.3 Feature-Based Query Operator

This operator is used to retrieve feature-based filtered regions of interest FR which can be used as an input by other operators that require regions of interest as an argument. It is possible to return a set of vertices that satisfy any kind of property that characterizes a feature, for instance, features computed using the *Voting Tensor* algorithm. Such operator can also select a buffer zone around

feature vertices by selecting a set of vertices that are not considered feature vertices on their own but are close enough to a minimum number of feature vertices.

We defined two versions of the feature-based filter operator: *fbqo* (R, d, k) which returns a region of interest composed of vertices that are features themselves or that contains within its neighborhood, defined by a distance d, at least k feature vertices; the *cfbqo*(R, d, k) which returns the complementary region to a feature-based filtered region. In our implementation, we used a topological distance instead of a geodesic distance but this would be easy to modify by using, for example, the Fast Marching Algorithm [16]. The resulting region must be pre-processed into an independent set if it is to be used as an input to the *subdivision operator*. We can define very powerful operators that rely on feature-based filtered regions because we can leverage feature information to modulate the desired deformations. Moreover, feature information can be propagated and attenuated when buffer zones are specified accordingly.

6.4 Geometric Operators

The geometric operators are usually used to produce deformations in medium to large scale. We define here a general geometric operator as *geom* (R) that takes as an input ta region of interest R given by a connected set of vertices. For the experiments, we implemented two specific instances of the geometric operator: the tapper operator *tpr* (R,r), where r is a non-uniform scaling and the twist operator, *twt* (R,θ) where θ is the twisting angle. In both operators, r and θ, respectively, are piecewise linear non-decreasing functions of one of the coordinates of the 3d embedding space [1].

Procedural Detail Operators. Consider a noise function $nf(x) : \mathbb{R}^3 \rightarrow \mathbb{R}$. The basic idea is to compute the value of a noise function at the coordinates $p(v)$ of a vertex $v \in V$ and use it as the intensity of the displacement of v in some direction $\boldsymbol{d}(v) : V \rightarrow \mathbb{R}^3$. Aiming to leverage the level of detail information, we modulate the local intensity of the deformation caused by the noise value $nf(p(v))$ via a function $I(lv, F) : \{0, \ldots, l_{max}\} \times \mathbb{R} \rightarrow \mathbb{R}^+$ where $lv(v) : V \rightarrow \{0, \ldots, l_{max}\}$ associates v to its level of detail and $F : V \rightarrow \mathbb{R}$ defines some measure computed on v, for instance, its Gaussian curvature $\kappa(v)$. Usually, $I(lv, F)$ is an attenuation function defined by a monotonic decreasing function of $lv(v)$. We define a general procedural detail operator as:

$$pdo(v, \boldsymbol{d}) = p(v) + I(lv(v), F(p(v))nf(p(v))\boldsymbol{d}(v) \tag{1}$$

A simple effect can be produced by considering $\boldsymbol{d}(v) = \boldsymbol{n}(v)$ where $\boldsymbol{n}(v)$ is the discrete estimate of the normal of v at M and $I = \frac{1}{lv(v)+1}$. This configuration produces stronger displacements at the finer components of the mesh. We mark a vertex as *noised* to avoid accumulating local deformations through the levels, but it is also possible to combine noise values generated in different levels to produce different effects.

We propose two instances of the procedural detail operator. In the first one, the noise function is based on the *Turbulence Function* proposed by Perlin [15] and the second one is based on the *Gabor Noise* [10] proposed by Lagae et al. The turbulence noise function, presented in Eq. 2, is a weighted sum of band pass noises. It can be controlled via four parameters: amplitude a, frequency fr, gain g and lacunarity lc. Lacunarity and gain define, respectively, the rate of change of the frequency and amplitude of the noise per octave and regulate the fractal behavior of the noise.

$$TN(v, a, lc, fr, g) = \sum_i a \left| \frac{(lc^i) * fr * p(v)}{g^i * a} \right| \tag{2}$$

Lacunarity usually takes values in $[1, 3]$ for a fractal behavior, whereas the gain is usually set to be the inverse of the lacunarity. Typical values are $fr = 2$, $lc = f$, $g = 0.5$ and $a = g$. The use of the previous values as arguments produces details where features align across scales so we must carefully modulate such parameters through the different levels of detail of the mesh. In our experiments, we defined the lacunarity and the gain in terms of the level of detail l respectively as $lc(l) = 1.1 + l * 0.1$ and $g(l) = 0.8/(l + 1)$. This configuration produced noise details that are smooth on large scales and rough on small scales. Our turbulence detail operator is represented as $tnpdo(v, a, lc, fr, g)$ defined by a generalization of Eq. 2 with the parameters a, lc and g depending on $lv(v)$.

Gabor noise [10] is a sparse convolution anisotropic noise based on a Gabor Kernel with accurate spectral control, which is able to provide a setup-free surface texturing. A band-pass Gabor noise presented in [10] is defined as:

$$GN(p, \dots) = \sum_i w_i g(K_i, a_i, F_{0,i}, \omega_{0,i}; p - p_i), \tag{3}$$

where w_i are the random weights, K represents the amplitude, a_i is the bandwidth, F_0 is the frequency and ω_0 is the orientation of the cosine in the kernel $g(p)$ as presented in Eq. 4:

$$g(p) = Ke^{-\pi a^2 |p|^2} cos\left[2\pi F_0 \left(p_x \cos\omega_0 + p_y \sin\omega_0\right)\right], \tag{4}$$

We can configure the parameters of the Gabor noise in a way similar to the setup of the Perlin noise. Also, the random positions p_i are distributed according to a Poisson process with mean λ.

The anisotropic feature of the Gabor Noise is paramount to the generation of detail aligned or guided by the features of a mesh as will be shown in the results section. To define a detail operator based on the Gabor Noise we can either modify the noise function directly or change the phase-augmented Gabor kernel. Perturbing the Gabor kernel can be done by regulating the K, a, ω variables, all at once or individually, according to the resolution level, which is similar to what is proposed for the turbulence and other functions. We can also modify the number of impulses per kernel according to this method, moving this value always as a power of two.

The idea is to find a nice relation between the many levels of resolution. Intuitively, for a better distribution, a higher number of impulses is preferred while manipulating the coarser levels, since the higher frequencies will be a dominant force. Also, this is a valuable tool for controlling which frequencies we want at each scale.

We define the Gabor Noise operator as $gno\ (v,\ l,\ \mathbf{K},\ \mathbf{a},\ \mathbf{F_0},\ \Omega,\ \mathbf{p})$ where $\mathbf{K}$, $\mathbf{a}$, $\mathbf{F_0}$, Ω, $\mathbf{p}$ are vectorial versions of the parameters in Eq. 3 that also depend on the level of detail of v.

6.5 Composite Operators

Composite operators produce more complex effects through the combination of two or more operators or by computing a procedure with such operators. We present two composite operators: *Variation Operator* and the *Organic Operator*.

The main purpose of the *variation operator* is to create detail variation across the surface of the object and through its many levels of resolution. This is achieved by modulating the frequency and the amplitude of the noise function, through each resolution level, in order to obtain the desired texture effect. It is basically an application of the procedural detail operator that relies on the 4-k mesh structure to use neighborhood relationships to propagate the noise from a vertex v to a neighbor vertex v' and increase or decrease its influence according to a topological distance $td(v, v')$ or geodesic distance $gd(v, v')$. To define the variation operator we use a variation of the procedural noise in Eq. 1 where the noise function nf is given by Perlin's *Turbulence Function* whose parameters are modulated by $lv(v)$, $I = \frac{1}{lv(v)+1}$ and α is the distance attenuation factor.

$$varo(v, v', \boldsymbol{d}, td, \alpha) = p(v) + \frac{I}{\alpha td(p(v), p'(v))}(\boldsymbol{d}(p(v))nf \tag{5}$$

The *organic operator* aims at creating organic looking structures. It relies on the *subdivision* and *smoothing operators* to create different shapes, but, for more general purposes, can be used without them.

This operator is composed of two steps, a global one, and a local one. The global step generates a deformation that produces an overall mesh detail texture, i.e. the patterns obtained through the noise functions presented before. The local step performs local manipulations creating more complex detail structures as a function of the level of detail.

Algorithm 1 describes the organic operator, which is used to create some of the images in the results section.

Algorithm 1 applies a deformation at a vertex v and propagates its effect to the vertex k-ring. Next, it performs a refinement on this new deformed region executing n steps of the subdivision operator. Finally, it deforms the refined area by choosing a random vertex v' within it and uses its normal to apply the procedural noise operator in v'. This will generate a wide area with an organic look because the vertices on the k-ring will be pushed outwards. We denote the *organic operator* as $orgo(R,l,k)$.

Data: Input region R, $k-ring$ defines the propagation neighborhood, and attenuation factor α
Result: Deformation of the refined region around vertices in R
for *Each* $v \in R$ **do**
  pdo$(v, \boldsymbol{n}(v))$;
  for *Each* $v' \in$ *the k-ring of* v **do**
    varo$(v, v', \boldsymbol{n}, td, \alpha)$
  end
  apply n **passes of:** sub(v);
  find random vertex v' in the refined area around v
  $\boldsymbol{d} \leftarrow \boldsymbol{n}(v')$;
  pdo$(v, \boldsymbol{d})$;
end

Algorithm 1. Creating extensions with the organic operator.

7 Results

Tests were performed in an Asus GT50vt notebook with an Intel Core 2 Duo CPU with 4 GB of RAM and a nVidia 9800M GTS GPU with 512 MB.

In Fig. 2, we present two ways to perform feature-based deformation. In the left figure, the deformation is restricted to the line feature area, whereas on the right we see it applied to a fan-disk centered at each vertex of the line features.

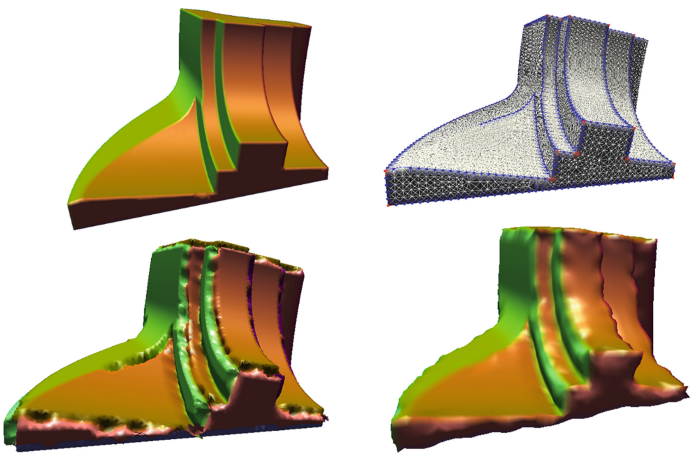

Fig. 2. Example of a deformation restricted to the feature line's area. In the bottom left we see an erosion on the fandisk across feature lines and, at the bottom right, a deformation to create a kneaded effect.

Figure 3 (left) illustrates an application of the organic operator. There are some steps necessary to generate this effect. First, we must select the region

of interest R disturb it with noise and then propagate the deformation across the 2-ring neighborhood area of each $v \in R$. We used the *variation operator* for this where the propagation is done in the following way: after moving the vertex in the direction of its normal we propagate the displacement value to the neighboring area with attenuation weight $\alpha = 1.5$ for the 1-ring and $\alpha = 2.0$ for the 2-ring vertices. The objective of this displacement is to create bumps with a smooth appearance. The next deformation is used to move the vertices in opposite directions creating a wide area at the top of the bump as shown in Fig. 3 (right). After this stage, the operator $tpndo(v, l, fr, a, lc = 6.0, g = 0.3)$ with default values of frequency and amplitude is used to create the idea of trees, bumpy ground and/or organic extensions.

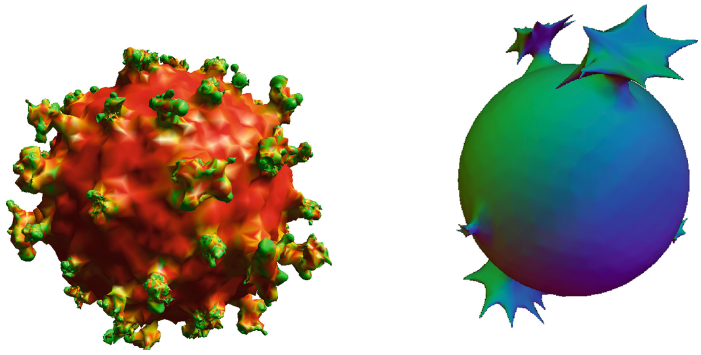

Fig. 3. (left) Example of an alien planet or virus cell using guided positioning. (right) Creating bumps with wide areas using variation operator.

In order to present a full view of our techniques pipeline, in Fig. 4 we show various steps of the same model being simplified, modified and refined. From left to right, starting with an original mesh (with roughly 40k vertices), we proceed to create a base mesh, through the mentioned simplification process, with approximately 1k vertices. The middle skull shows how the removed vertices are mapped over each face (triangle) the model. Next, a deformation is performed, in this specific case a twist operator, and a reparameterization step is executed to guarantee the consistency of the mesh needed during the refinement process. Finally, the last skull presents the model after all possible refinement steps, and how the deformation is correctly propagated from the base mesh to the higher levels of detail.

Figure 5 (right) illustrates a complete example where several operators are combined for a final modeling effect. We use as an input a skull mesh Fig. 5 (left) with its mean curvature calculated. It is easy to spot the high curvature areas around the eyes, nose, and teeth. In this particular example, we want to create a beard effect, without deforming the area of the teeth, and also to give a hair appearance to this bald model.

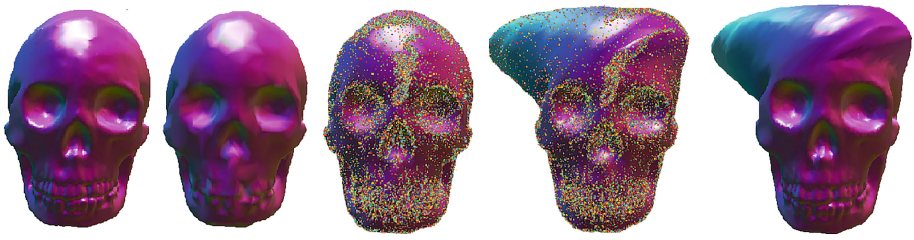

Fig. 4. Steps of our method from left to right: input mesh (˜40k vertices), base mesh (˜1k vertices) created through the simplification process, parameterization mapping, deformation and remapping, and, finally, the result of the refinement proces

We first start with the *tnpdo* operator applied to the vertices v in a ROI filtered by curvature value selecting only those of higher curvature and y value below the nose region. This introduces a random displacement characterizing the beard. In the second step, we apply only at the coarsest level, the *tapper operator (*tpr) in the x and y coordinates while maintaining z untouched. After a few steps of refinement and deformation propagation, we apply the *twist operator tst* in the region of higher z coordinate to create a curl effect followed by a *gaussian noise operator gdo* in the same region that creates details aligned with the z direction, producing the hair movement.



Fig. 5. (left) Original mesh. (right) Deformation of hair and beard.

8 Conclusion

This work proposes a new framework for adaptive multiresolution mesh detail addition and deformation. These are performed on a 3D shape represented by an adaptive parameterized variable resolution mesh representation together with a set of deformation and auxiliary operators. Our modeling pipeline enables deformation and detail addition at any level of the representation. Also, it does not build a geometrical multiresolution representation. Instead, it uses a topological multiresolution representation based on feature-sensitive simplification.

In such representation, features such as strong edges, weak edges and corners, when present in the original shape, are kept even at the coarsest levels enabling the use of such elements as control elements and input to deformation or detail addition. We argue that features are important elements for mesh edition operation and must be made available for manipulation whenever possible.

Operators are devised to work with a generalized deformation handle which includes regions of interest, its corresponding levels of detail and a directional vector field that characterizes the direction of propagation of the deformation. Differently from other similar works we do not rely on remeshing of the original mesh and also do not store sequences of operation performed at the adaptive representation.

Different experimental results have shown that our method is able to produce quite promising results especially considering controlled procedural detail addition. The possibility of adding detail aligned with features and specific directions on the mesh is one of the most powerful effects we presented. As a result, there are many possibilities for future research.

In this work, the focus is on the representation, operator definition and control specification. Hence, implementation of the method up to now is still not optimized to enable interactive manipulation. We intend to pursue, in the future, interactive rates as well as ways of converting friendly user-interactive mechanisms into sequences of deformation operators that take as input automatically specified general deformation handles.

In the presented method, controlling the procedural detail scale according to the level of detail of the mesh requires an artificial discretization of the signals scale as our representation does not build a continuum of mesh representations. Also, looking for ways of describing a continuous version of our representation as in geometrical multiresolution, but keeping the most important features of the mesh at coarser levels is a major goal. This would make possible to leverage all the possibilities of our method and at the same time achieve a better syntonization of the levels of detail with the scales of procedural signals.

References

1. Barr, A.H.: Global and local deformations of solid primitives. In: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1984, pp. 21–30. ACM, New York (1984)
2. Bendels, G.H., Klein, R., Schilling, A.: Image and 3d-object editing with precisely specified editing regions. In: Ertl, T., et al. (eds.) Vision, Modeling and Visualisation 2003, pp. 451–460. Akademische Verlagsgesellschaft Aka GmbH, Heidelberg (2003)
3. Biermann, H., Martin, I.M., Zorin, D., Bernardini, F.: Sharp features on multiresolution subdivision surfaces. Graph. Models **64**(2), 61–77 (2002)
4. Botsch, M., Pauly, M., Rossl, C., Bischoff, S., Kobbelt, L.: Geometric modeling based on triangle meshes. In: ACM SIGGRAPH 2006 Courses, SIGGRAPH 2006. ACM, New York (2006). https://doi.org/10.1145/1185657.1185839
5. Dekkers, E., Kobbelt, L.: Geometry seam carving. Comput. Aided Des. **46**, 120–128 (2014)

6. Derzapf, E., Grund, N., Guthe, M.: Parallel progressive mesh editing. In: Eurographics Symposium on Parallel Graphics and Visualization, Swansea, Wales, UK, 2014, Proceedings, pp. 33–40 (2014). https://doi.org/10.2312/pgv.20141082
7. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary meshes. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995, pp. 173–182. ACM, New York (1995)
8. Hoppe, H.: Progressive meshes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, pp. 99–108. ACM, New York (1996). https://doi.org/10.1145/237170.237216
9. Kobbelt, L., Stamminger, M., Seidel, H.P.: Using subdivision on hierarchical data to reconstruct radiosity distribution **16**(3), C347–C355 (1997). https://doi.org/10.1111/1467-8659.16.3conferenceissue.36, Proceedings Eurographics 1997
10. Lagae, A., Lefebvre, S., Drettakis, G., Dutré, P.: Procedural noise using sparse gabor convolution. In: ACM SIGGRAPH 2009 Papers, SIGGRAPH 2009, pp. 54:1–54:10. ACM, New York (2009)
11. Lee, A.W.F., Sweldens, W., Schröder, P., Cowsar, L., Dobkin, D.: Maps: multiresolution adaptive parameterization of surfaces. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, pp. 95–104. ACM, New York (1998). https://doi.org/10.1145/280814.280828
12. Loop, C.: Smooth Subdivision Surfaces Based on Triangles. Department of Mathematics, University of Utah (1987)
13. Maximo, A., Velho, L., Siqueira, M.: Adaptive multi-chart and multiresolution mesh representation. Comput. Graph. **38**, 332–340 (2014). https://doi.org/10.1016/j.cag.2013.11.013
14. Medioni, G., Tang, C.K., Lee, M.S.: Tensor voting: theory and applications. In: Proceedings of RFIA (2000)
15. Perlin, K.: An image synthesizer. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1985, pp. 287–296. ACM, New York (1985)
16. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. In: Proceedings National Academy of Sciences, pp. 1591–1595 (1995)
17. Sorkine, O., Botsch, M.: Tutorial: interactive shape modeling and deformation. In: EUROGRAPHICS (2009)
18. Taubin, G.: A signal processing approach to fair surface design. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995, pp. 351–358. ACM, New York (1995). https://doi.org/10.1145/218380.218473
19. Velho, L.: Mesh simplification using four-face clusters. In: Shape Modeling International, pp. 200–208. IEEE Computer Society (2001)
20. Velho, L., Gomes, J.: Variable resolution 4-k meshes: concepts and applications. Comput. Graph. Forum **19**(4), 195–212 (2000)
21. Velho, L., Perlin, K., Biermann, H., Ying, L.: Algorithmic shape modeling with subdivision surfaces. Comput. Graph. **26**(6), 865–875 (2002)
22. Wang, S., Hou, T., Su, Z., Qin, H.: Diffusion tensor weighted harmonic fields for feature classification, pp. 93–98 (2011)
23. Yümer, M.E., Chaudhuri, S., Hodgins, J.K., Kara, L.B.: Semantic shape editing using deformation handles. ACM Trans. Graph. **34**(4), 86 (2015)
24. Zorin, D., Schröder, P.: Subdivision for Modeling and Animation. Technical report, SIGGRAPH 2000 (2000). Course Notes