Non-uniform Differential Mesh Deformation

Dong Xu[1], Hongxin Zhang[1], and Hujun Bao[1]

State Key Laboratory of CAD&CG, Zhejiang University, P.R.China,
{xudong, zhx, bao}@cad.zju.edu.cn

Abstract. In this paper, we propose a novel mesh deformation approach via manipulating differential properties non-uniformly. Guided by user-specified material properties, our method can deform the surface mesh in a non-uniform way while previous deformation techniques are mainly designed for uniform materials. The non-uniform deformation is achieved by material-dependent gradient field manipulation and Poisson-based reconstruction. Comparing with previous material-oblivious deformation techniques, our method supplies finer control of the deformation process and can generate more realistic results. We propose a novel detail representation that transforms geometric details between successive surface levels as a combination of dihedral angles and barycentric coordinates. This detail representation is similarity-invariant and fully compatible with material properties. Based on these two methods, we implement a multiresolution deformation tool, which allow the user to edit a mesh inside a hierarchy in a material-aware manner. We demonstrate the effectiveness and robustness of our methods by several examples with real-world data.

1 Introduction

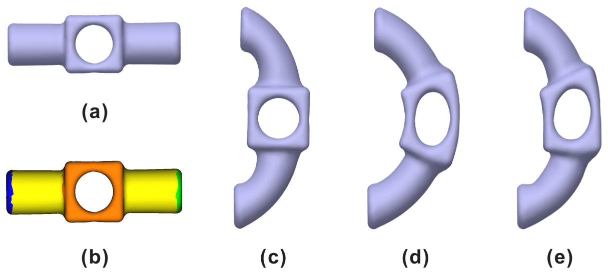

Fig. 1. Mesh deformation with non-uniform control. (a) is the original model; (b) is the color plot of user-specified materials, the green region is the handle and the blue region is the constraint; (c) and (d) are results generated with and without non-uniform control respectively; (e) is the result generated with the non-uniform propagation but with the uniform reconstruction.

Mesh deformation has been widely studied in computer graphics. Most existing techniques, such as freeform deformations, multiresolution techniques and recently introduced differential domain methods are mainly designed to propagate the deformation imposed by the user evenly into the influence region uniformly. However, real world object often contains parts with different materials. Given outside forces, for a certain part how it deforms is determined by its corresponding material properties. Using material-oblivious deformation techniques to edit objects with non-uniform material properties often produces implausible results with unnatural shape artifacts.

In this paper, we present a novel mesh deformation technique that allows the surface mesh to be deformed in a non-uniform way. It is based on material-dependent Poisson equations and supplies non-uniform controls in both the propagation process and the reconstruction process. The user is involved into the deformation process by setting material properties to indicate non-uniform regions while the rest takes the default value. These user-specified material properties guide the propagation of local transformations as well as the reconstruction to absolute coordinates. Our method inherits the advantages of differential domain methods and provides more flexible control of the deformation process. In particular, surface details can be well preserved during the deformation, and in a material-aware manner.

To facilitate the editing of large meshes, we further extend our non-uniform deformation technique into a multiresolution version. Multiresolution techniques have been proved to be powerful to process gigantic models. However, lack of material-dependent mechanism prevents existing multiresolution deformation approaches to be adapted for our purpose. Instead we propose a novel detail representation that transforms geometric details between successive surface levels as a combination of dihedral angles and barycentric coordinates. This detail representation is similarity-invariant while existing representations, such as local frame displacements [1–3] and displacement volumes [4], are rigid-invariant. More importantly, the similarity-invariant detail representation is fully compatible with material properties. Based on it, the user can edit the simplified base mesh with our single-resolution non-uniform editor and obtain the detailed result automatically.

The rest of this paper is organized as follows. After briefly reviewing the prior arts with a focus on multiresolution techniques and differential domain methods in Section 2, we present the non-uniform deformation technique in Section 3. In Section 4 we elaborate how to perform non-uniform deformations in the multiresolution context. We demonstrate that more realistic results can be generated with the help of material properties in Section 5. Finally, we draw conclusions and point out possible future work in Section 6.

2 Related Work

Our approach builds on recently introduced differential domain methods [5] that represents surface details as differential properties. These approaches manipulate differential properties and reconstruct vertex coordinates via solving a sparse linear system. They have a valuable feature that geometric details can be well-preserved during the editing process. Since differential properties are only translation-invariant, they must be properly transformed according to user interactions. The determination of local transformations can be achieved by either explicit interpolation [6–8] or implicit fitting [9].

Zhou et al. [10] extend the Laplacian coordinates to the volumetric graph to address problems with large mesh deformations. However, all of these approaches regard the edited mesh to be made up of a uniform material, thus lacking of additional control of the deformation process. Based on this observation, we extend differential domain methods to supply position-dependent control by incorporating user-specified material properties.

In the context of boundary constraint modeling, Botsch and Kobbelt [11] propose a freeform modeling framework based on a family of linear differential equations. They point out that the deformation in certain direction can be enhanced by explicitly shrinking the underlying parameter domain in the same direction, which affects the discretization of Laplacian operator consequently. Our method achieves more general control of the deformation by allowing the user to specify per-face material properties, which implicitly modifying the domain mesh in a non-linear manner. Material properties have also been considered by Popa et al. [12] to control the propagation of local transformations. Our method differs in the way that we also consider material properties in the reconstruction process (See Figure 1).

Multiresolution technique has been introduced into computer graphics community for more than ten years [13]. Some approaches depend on semi-regular meshes [1] while others work on irregular meshes directly [2, 3, 11]. Most multiresolution editing techniques manipulate a static surface hierarchy, but vertex connectivity of the base surface [14] or the hierarchy itself [15] can also be dynamically rearranged during large deformations. These approaches share a common aspect that geometric details between successive levels are encoded as local frame displacements. These displacements can be scalars along the normal directions [16] or vectors [1–3]. Since local frame displacements are handled individually, the reconstructed detailed surface may have unnatural volume changes when the base surface endures large deformations. Consequently, Botsch and Kobbelt [4] propose to use displacement volumes [4] instead. Displacement volumes are kept locally constant during the reconstruction process to preserve volume and avoid local self-intersections. However, both local frame displacements and local volumes do not support material-dependent reconstructions, making us exploring a new detail representation.

3 Mesh Editing with Non-uniform Control

Previous differential domain methods deform surfaces in a material-oblivious way. In this section, we present how to enhance these techniques by incorporating user specified non-uniform materials. With the non-uniform control mechanism, the deformation process can be tuned in a finer granularity than previous differential domain methods. Therefore, more realistic results can be generated easily.

3.1 Material-Dependent Poisson Equation

A simple form of Poisson equation have been successfully applied to the context of mesh editing [6]. In physics a more general form of this elliptic equation is used to describe the phenomena of steady-state heat conduction in a 3D solid medium. Commonly, the exact form in 3D Euclidian space is

$$\Delta_\kappa T = \frac{\partial}{\partial x}\left(\kappa_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa_y \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\kappa_z \frac{\partial T}{\partial z}\right) = -q_v. \tag{1}$$

Here T is a steady-state temperature field, q_v is the source term in the interior. And κ_x, κ_y and κ_z are speed functions, namely thermal conductivities along three axial directions. Therefore, different solid medium result different steady-state temperature fields even under the same set of boundary conditions. This basic observation motivates us to use material-dependent control for differential mesh editing.

In this paper we only consider isotropic materials, i.e, $\kappa_x = \kappa_y = \kappa_z = \kappa$. When κ is constant, Eq. (1) describes a uniform material case which is applied in [6]. In the following we explore the Poisson equation defined on surfaces with non-uniform materials, and the thermal conductivity κ is a scalar field on manifold surface.

Since we adopt triangle meshes as the underlying surface representation, we have to discretize material-dependent differential operators on 2-manifold meshes. For this purpose, we first briefly review the uniform case, i.e., the standard differential operators used in [6]. Given a piecewise linear scalar field $f(\mathbf{v}) = f_i\phi_i(\mathbf{v})$ defined on a 3D mesh, the gradient operator is defined as $\nabla f(\mathbf{v}) = f_i\nabla\phi_i(\mathbf{v})$, where f_i is the scalar value on vertex $\mathbf{v_i}$, $\phi_i(\mathbf{v})$ is the piecewise linear basis and $\nabla\phi_i(\mathbf{v})$ is its gradient. Given a piecewise constant vector field $\mathbf{w}$, the divergence of the vector field $\mathbf{w}$ is defined as

$$\nabla \cdot \mathbf{w}(\mathbf{v}_i) = \sum_{T \in N_T(\mathbf{v}_i)} A_T \nabla\phi_i^T \cdot \mathbf{w}, \tag{2}$$

where $N_T(\mathbf{v}_i)$ is the adjacent triangle set of the vertex $\mathbf{v_i}$ and A_T is the area of the triangle T. Combining the gradient operator and the divergence operator, we get the Laplacian operator

$$\Delta f(\mathbf{v}_i) = \frac{1}{2} \sum_{j \in N_\mathbf{v}(\mathbf{v}_i)} (\cot\alpha_{ij} + \cot\beta_{ij})(f_i - f_j), \tag{3}$$

where $N_\mathbf{v}(\mathbf{v}_i)$ is the adjacent vertex set of the vertex $\mathbf{v}_i$, α_{ij} and β_{ij} are two opposite angles of the edge $(\mathbf{v}_i, \mathbf{v}_j)$.

Now we extend the above procedure to the material-dependent case. We assume that the material property κ is a piecewise constant function i.e., $\kappa \equiv \kappa_T$ in a triangle T. Note that the thermal conductivity terms are attached after first partial differential operator in Eq.1. We thus define material-dependent gradient of basis $\phi_i(\cdot)$ as $\kappa_T\nabla\phi_i^T$. In other words, the gradient of the pieces linear basis is resized according to its material property. Then we can represent the material-dependent divergence operator as

$$\nabla_\kappa \cdot \mathbf{w}(\mathbf{v}_i) = \sum_{T \in N_T(\mathbf{v}_i)} \kappa_T A_T \nabla\phi_i^T \cdot \mathbf{w}, \tag{4}$$

and the material-dependent Laplacian operator as

$$\Delta_\kappa f(\mathbf{v}_i) = \frac{1}{2} \sum_{j \in N_\mathbf{v}(\mathbf{v}_i)} (\kappa_{j-1}^2 \cot \alpha_{ij} + \kappa_j^2 \cot \beta_{ij})(f_i - f_j). \tag{5}$$

Given the guidance field $\mathbf{w}$ and boundary conditions $f_i = f_i^*, \mathbf{v}_i \in \partial\Omega$, we get the material-dependent Poisson equations:

$$\Delta_\kappa f = \nabla_\kappa \cdot \mathbf{w}. \tag{6}$$

3.2 Non-uniform Propagation

To ensure visually desirable deformation results, local transformations imposed by user interactions must be propagated (weighted with a fall-off function) into the region of interest (ROI) smoothly. Inspired by [7], we consider the material-dependent propagation as a analogy of the heat conduction in a non-uniform medium. Here, material properties are interpreted as their thermal conductivity. The scalar field function f guided the propagation process can be computed by the following material-dependent Laplace equation:

$$\Delta_\kappa f = 0, \tag{7}$$

where Δ_κ is material-dependent Laplacian operator (See Eq. 5).

Actually, the propagation field f is equivalent to the steady-state temperature field with boundary temperatures set to be 1 on handle vertices and 0 on constrained vertices. f is also the minimizer of the following energy function:

$$\min_f \int_\Omega \|\kappa(\omega)\nabla f\|^2 d\omega, \tag{8}$$

where $\kappa(\omega)$ is the user-specified material property as thermal conductivity. Intuitively, if the user wants to keep some regions as rigid as possible, he/she can set the material properties of these regions with a large value. On the contrary, if the user expects certain regions to be freely deformed, he/she can set them with a small value.

After the propagation field f is solved, we use it as the fall-off function to weight local transformations. For rotation transformation, we use f to multiply the rotation angle while for scaling transformation, we adopt f to linearly interpolate between the scaling ratio r and 1 (no scale). Then we combine these local transformations together according to the user-selected transformation option.

3.3 Non-uniform Reconstruction

The propagation process assigns a local transformation to each triangle and we obtain guidance vectors by applying it to original gradient vectors. Unlike [6], we consider material properties in the reconstruction process as well. The Poisson mesh solver used in [6] can be regarded as a special case of our material-dependent one. Our method is equivalent to the minimization of the following energy:

$$\min_{f} \int_{\Omega} \kappa^2(\omega) \left\| \nabla f - \mathbf{w} \right\|^2 d\omega. \tag{9}$$

Note that the material-dependent Laplacian operator (cf. Eq. 5) defined on a domain mesh M with a non-uniform material can be regarded as a standard one (cf. Eq. 3) defined on another domain mesh M' with a uniform material. The relationship between M and M' lies in for each edge $(\mathbf{v}_i, \mathbf{v}_j)$, the following equation is satisfied:

$$\kappa_{j-1}^2 \cot \alpha_{ij} + \kappa_j^2 \cot \beta_{ij} = \cot \alpha'_{ij} + \cot \beta'_{ij}. \tag{10}$$

From this aspect, we can think of that material properties act as the modifying factors of the original domain mesh M.

3.4 The User Interface

We adopt the handle-based editing metaphor. During editing, the user selects the region of interest (ROI) and deforms the mesh by manipulating a small region inside the ROI, called handle. The user manipulates the handle with a 9 degree-of-freedom manipulator. Besides this editing interface, we design a simple pick-and-drag interface that only takes the pure translation of the handle as the input. In addition, user can paint different parts of ROI with different colors that represent corresponding materials. After the user drags the handle, our method automatically induces the necessary rotation and scaling for the ROI as well as the handle itself.

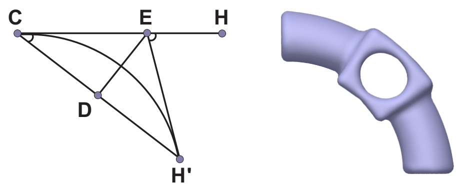

Fig. 2. Illustration of the determination of local transformations from the handle movement.

The basic idea of determining the rotation and scaling from the handle movement comes from the Hermite interpolation, as shown in Figure 2. Let C denotes the center of the boundary connecting constrained vertices and free vertices, H denotes the center of handle vertices and H' denotes the new center of handle vertices after translation. Note that the three points C, H and H' can uniquely determine a plane provided they are not degenerate. The rotation axis $\mathbf{a}$ can be easily determined by the cross product of two vectors $\overrightarrow{HC}$ and $\overrightarrow{H'C}$. The scaling factor s is defined to be the ratio between the length of $\overrightarrow{H^{'}C}$ and that of $\overrightarrow{HC}$. The left problem is to define the rotation angle. We consider the circle passing through two points C and $H^{'}$ and tangent to the vector $\overrightarrow{HC}$ at the point C. Then, we define the rotation angle θ to be $\angle(HEH^{'})$, where the point E is the intersection of the line $\overrightarrow{HC}$ and the perpendicular bisector of the line $\overrightarrow{H^{'}C}$. Actually, the angle θ is twice the angle $\angle(HCH^{'})$. Therefore, we do not need to construct the circle at all. Provided the three points C, H and $H^{'}$ are coplanar, we simply ignore the rotation transformation. We supply several options to support different combination of these local transformations. These estimated local transformations are propagated into the ROI (See Section 3.2) to generate guidance fields and the deformed surface mesh is reconstructed with Poisson equations (See Section 3.3).

4 Multiresolution Non-uniform Editing

The multiresolution paradigm is an efficient way to deforming large meshes with complex geometric details. Typically, a multiresolution editing framework consists of three major components - the decomposition component, the reconstruction component and the deformation component. Since the non-uniform control of mesh deformation is the focus of this paper, in this section we show how to achieve this goal in the multiresolution scenario. Note that the decomposition component is independent to material properties as a pre-processing step while the rest two are material-dependent.

4.1 Mesh Decomposition and Detail Encoding

Aiming at editing large meshes, we employ the Progressive Mesh (PM) [17] to represent the surface hierarchy. A PM is created by recursively applying edge-collapse operations to a detailed input mesh. In a PM representation, the edge-collapse and the vertex-split are atomic operations for the decomposition component and the reconstruction component respectively. Note that in both operations, only the central one or two vertices' coordinates are modified while all the rest do not change their position. This observation is particularly important since it allows us to localize the detail encoding and decoding procedures only with respect to the local stencil of a given edge.

After the surface hierarchy is created, geometric details between successive levels need to be encoded. Geometric details are typically defined as the difference between the original geometry and the approximated smoothed geometry. We consider the membrane surface as the smoothed approximation, which can be obtained by solving a Laplace equation defined on the local stencil of the given edge e, denoted as $L(e)$ with Dirichlet boundary conditions given by vertex coordinates on the boundary $\partial L(e)$. Since only two free vertices in the local stencil, the corresponding Laplace equation is a linear system with two equations:

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}, \tag{11}$$

where a, b, c come from Eqn. 3 and $\mathbf{u}_1$ and $\mathbf{u}_2$ are boundary conditions.

We adopt the original geometry to define the Laplacian operator so that we can smooth the geometry without affecting the underlying parameterization [18]. The corresponding weights used to define the Laplacian operator can be stored or computed on the fly, trading off speed versus memory. After the smoothed approximation is solved, we define the detail coefficients between a pair of triangles coming from the original geometry and the smoothed approximation respectively (see Figure 3).

Generally speaking, locating one detail triangle $T_1 = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ with respect to the corresponding base triangle $T_2 = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ can be decomposed into two steps, which are summarized by nine independent parameters $(x, y, z, \theta_1, \theta_2, \alpha_1, \beta_1, \alpha_2, \beta_2)$ (See Figure 3). The first step aligns the vertex $\mathbf{w}_1$ to the vertex $\mathbf{v}_1$ and the offset vector is (x, y, z). The second step rotates the triangle T_1 along the axis dir defined by the cross product of the normal vector $\mathbf{n}_2$ and $\mathbf{n}_1$ so that both triangles are co-planar. The rotation angle θ_1 is equivalent to the dihedral angle between the two triangles. For the sake of the reconstruction process, we need to encode the axis dir with respect to the base triangle as well. Since the vector dir is co-planar with the base triangle T_2, it can be located by rotating the vector $\mathbf{v}_2 - \mathbf{v}_3$ around the axis $\mathbf{n}_2$ with a rotation angle θ_2. After the second step, we get the rotated detail triangle $T_1^{'} = (\mathbf{w}_1^{'}, \mathbf{w}_2^{'}, \mathbf{w}_3^{'})$ that lies in the same plane with triangle T_2. Now we can record the coordinates of vertices $\mathbf{w}_2^{'}$ and $\mathbf{w}_3^{'}$ with respect to the triangle $T_2 = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and results in the rest four barycentric coordinates $(\alpha_1, \beta_1, \alpha_2, \beta_2)$.

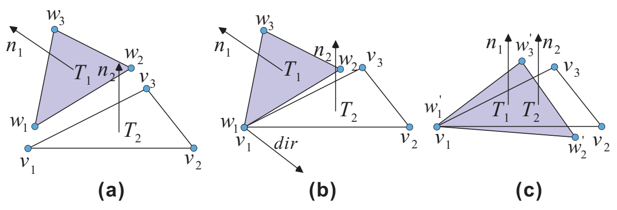

Fig. 3. Encoding a detail triangle T_1 w.r.t. a base triangle T_2.

Actually, since our reconstruction method can automatically determine the position of the detailed triangle from the base one, we do not need to record the offset vector (x, y, z). Therefore, our similarity-invariant detail coefficients consist of the rest six parameters $(\theta_1, \theta_2, \alpha_1, \beta_1, \alpha_2, \beta_2)$.

4.2 Detail Reconstruction

After the user performs a material-dependent deformation on a base mesh, we need to reconstruct pre-recorded geometric details with respect to user-specified material properties. There are two issues concerning material-dependent detail reconstruction process. The first one is that material properties specified on the low-resolution mesh should be up-sampled. The second one is each refinement step should take the (up-sampled) material properties into account. We present our solutions in the following paragraphs in details.

Specifically, after a vertex-split operation is performed, we immediately up-sample the material properties. The material property of a newly-split triangle is set to be the weighted average of its adjacent triangles. We adopt the invert-distance as the weighting scheme. Then, a three-step reconstruction process is employed to reconstruct geometric details from previously encoded detail coefficients. The first step is to generate the approximated smoothed geometry by material-dependent Laplace equation(Eqn. 11) with new Dirichlet boundary conditions given by the deformed base surface. Now we can retrieve the detail triangles from the corresponding base triangles one-by-one. In fact, the second step is carried out in the reverse order of the encoding process. First, we determine the intermediate detail triangle $T_1^{'} = (\mathbf{w}_1^{'}, \mathbf{w}_2^{'}, \mathbf{w}_3^{'})$ using the barycentric coordinates $(\alpha_1, \beta_1, \alpha_2, \beta_2)$ with respect to the base triangle $T_2 = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$. Note that the vertex $\mathbf{w}_1$ is superposed on the vertex $\mathbf{v}_1$. Then, the axis dir is obtained by rotating the vector $\mathbf{v}_2 - \mathbf{v}_3$ around the axis $\mathbf{n}_2$ with the rotation angle θ_2. Finally, the intermediate detail triangle $T_1^{'}$ is rotated around the axis dir with the rotation angle $-\theta_1$, resulting in the detail triangle T_1.

Since the detail triangles have been extracted from the corresponding base triangles independently, the third step is to glue them together and generate the consistent vertex position for the central two vertices come from the vertex-split operation. To serve for the purpose, a local material-dependent Poisson equation is employed. We gather gradient vectors from broken detail triangles as the guidance field, which determines the vertex position together with the new boundary conditions:

$$\begin{bmatrix} a_\kappa & b_\kappa \\ b_\kappa & c_\kappa \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^{'} + \mathbf{w}_1 \\ \mathbf{u}_2^{'} + \mathbf{w}_2 \end{bmatrix}, \tag{12}$$

where a_κ, b_κ and c_κ come from Eqn. 5, $\mathbf{u}_1^{'}$ and $\mathbf{u}_2^{'}$ are new boundary conditions, $\mathbf{w}_1$ and $\mathbf{w}_2$ are the material-dependent divergence of vertex $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively.

Although multiresolution techniques [3] have been employed in the Poisson-based mesh solver [6] for acceleration, our framework differs in the way that it can automatically adapt geometric details to a scaled base surface via incorporating similarity-invariant detail representation. Moreover, our detail reconstruction method is particularly suitable to material-dependent multiresolution editing while previous methods are generally difficult for this purpose.

5 Results and Discussions

Based on techniques presented in Section 3 and Section 4, we implement a multiresolution editing tool for surface meshes. It can work on the single-resolution mode as well as the multi-resolution mode. Our editing tool can run interactively for moderate models with around 20k vertices in the single-resolution mode. In the multi-resolution mode, comparing with local frame displacements, our material-aware detail-reconstruction runs about 10-20% slower.

When editing CAD models, material properties can help certain feature regions to be better preserved. Figure 1(c) demonstrates such a task. The central feature region of the Mechpart model need to be preserved during the resizing of the main body. We achieve this goal by painting these feature regions with a large material valued 5 and perform non-uniform deformation with our technique.

Changing the value of material properties can lead to different deformation effects under the same user interaction. Figure 4 demonstrates several results obtained with different material settings. The default value of material properties is set to 1 and we have found the range $[1,5]$ is enough to simulate most of real world material-dependent deformations. We incrementally paint material properties on the crus part and the foot part with the value 2, and on the thigh part with the value 5. At the same time, we get more and more realistic results as shown in Figure 4 (b), (c) and (d).

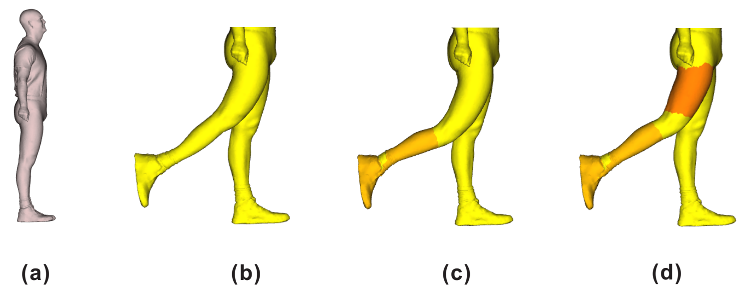

Fig. 4. Deforming the right leg of the Man model with different material settings.

Figure 5 illustrates and compares results obtained with different detail representations. We simplify the Elephant model (Figure 5 (a)) and reconstruct geometric details from a uniformly shrunken base mesh (Figure 5 (b)). This experiment is fairly simple, but we can clearly distinguish the difference between the result generated with our method (Figure 5 (c)) and that with local frame displacement (Figure 5 (d)). Note that fine details around the elephant's ear are not well-preserved in Figure 5(d) due to the lack of adaptation to arbitrary scaling. We also try to do some experiments, helping local frame displacements to adjust automatically. One such attempt is to resize displacements according the square root of the ratio between the area of the original base mesh and that of the deformed one and we call it *the global adjustment*. As shown in Figure 5 (e), we note the Elephant's ear is still disturbed. The other attempt is similar to the first one, but the area considered here is confined to the local stencil and that is the reason for its name - *the local adjustment*. However, this modification does not work as well (Figure 5 (f)). Figure 5 (c), (d), (e) and (f) are zoomed in two times for better visualization.

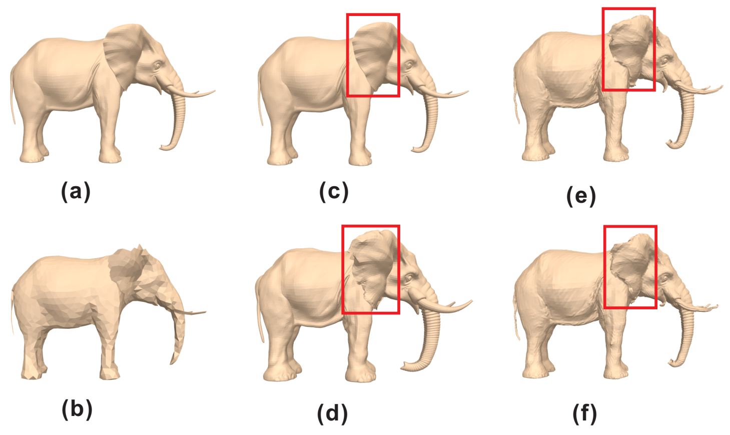

Fig. 5. The ability of similarity-invariance is over-looked in existing detail representations.

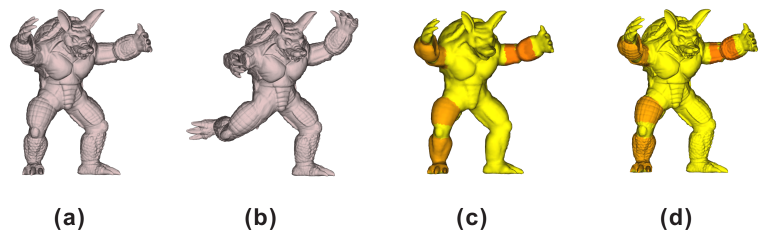

Fig. 6. After applying three non-uniform deformations, the Armadillo model (a) is now ready to kick a soccer (b). (d) is the visualization of material properties on the detailed mesh, which are automatically up-sampled from material properties specified by the user on the base mesh (c).

Figure 6 shows the Armadillo model kicking a soccer, which is generated with the multiresolution editing mode. We decimate the original model (170k vertices) to a simplified version (15k vertices). We perform three non-uniform edits on the both hands and the right leg of the simplified model. The detailed edited version is automatically reconstructed by our tool with the help of similarity detail coefficients and material properties. See our video submission for the whole editing process.

6 Conclusions and Future Work

In this paper, we propose a novel technique to deform the surface mesh non-uniformly by incorporating user-specified material properties. This goal is achieved by overloading previous material-independent discrete differential operators and Poisson equations. Moreover, we allow multiresolution mesh editing in a material-aware manner by incorporating a novel similarity-invariant detail representation. Several real world examples demonstrate that plausible material-dependent deformation results can be generated by our method easily. As pointed out in Section 3.3, designing a tailored domain mesh for a specific deformation task is a valuable research direction.

7 Acknowledgement

This work is supported in part by the National Basic 973 Program of China (Grant No. 2002CB312102) and the National Natural Science Foundation of China (Grant Nos. 60021201 and 60505001).

References

1. Zorin, D., Schröder, P., Sweldens, W.: Interactive multiresolution mesh editing. In: Proceedings of ACM SIGGRAPH 1997, ACM Press (1997) 259–268
2. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P.: Interactive multi-resolution modeling on arbitrary meshes. In: Proceedings of ACM SIGGRAPH 1998, ACM Press (1998) 105–114
3. Guskov, I., Sweldens, W., Schröder, P.: Multiresolution signal processing for meshes. In: Proceedings of ACM SIGGRAPH 1999, ACM Press (1999) 325–334
4. Botsch, M., Kobbelt, L.: Multiresolution surface representation based on displacement volumes. Computer Graphics Forum (Eurographics 2003) **22**(3) (2003) 483–491
5. Sorkine, O.: Laplacian mesh processing. In: Eurographics 2005 STAR report. (2005)
6. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. **23**(3) (2004) 644–651
7. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. Comput. Graph. Forum **24**(3) (2005) 601–609
8. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. **24**(3) (2005) 479–487
9. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Symposium on Geometry Processing. (2004) 179–188
10. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph laplacian. ACM Trans. Graph. **24**(3) (2005) 496–503
11. Botsch, M., Kobbelt, L.: An intuitive framework for real-time freeform modeling. ACM Trans. Graph. **23**(3) (2004) 630–634
12. Popa, T., Julius, D., Sheffer, A.: Material aware mesh deformations. In: Proceedings of ACM SIGGRAPH 2005. (2005) Posters.
13. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary meshes. In: Proceedings of ACM SIGGRAPH 1995. (1995) 173–182
14. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: Symposium on Geometry Processing. (2004) 189–196
15. Kobbelt, L., Bareuther, T., Seidel, H.P.: Multiresolution shape deformations for meshes with dynamic vertex connectivity. Comput. Graph. Forum **19**(3) (2000)
16. Guskov, I., Vidimce, K., Sweldens, W., Schröder, P.: Normal meshes. In: Proceedings of ACM SIGGRAPH 2000. (2000) 95–102
17. Hoppe, H.: Progressive meshes. In: Proceedings of ACM SIGGRAPH 1996. (1996) 99–108
18. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of ACM SIGGRAPH 1999. (1999) 317–324