Material Aware Mesh Deformations

Tiberiu Popa[*]
University of British Columbia

Dan Julius[†]
University of British Columbia

Alla Sheffer[‡]
University of British Columbia

Abstract

Most real life objects consist of non-uniform materials. Therefore, under deformation, the amount of bending and shearing across different parts of the objects varies based on the material stiffness in those regions. In the virtual environment there are three prevalent approaches to model deformation: purely geometric, skeleton based, and physically driven. The first approach typically considers only model geometry, treating objects as if they consist of uniform material. The skeleton based approach is applicable to only a subset of the models and typically supports only binary gradation of material stiffness. Lastly, the physics based approach successfully supports smooth material variation throughout the models, but is typically quite time-consuming. It is also purely physics driven, thus allowing little user control on the results.

This paper proposes a new approach to model deformation incorporating material properties into the geometric deformation setting. Our new approach combines the ease and efficiency of geometric model editing with support of non-uniform material properties, including user defined stiffness controls for both bending and shearing. It provides higher level of material variation (need better term) than possible using traditional skeletons and can be used for models of any shape, from animals to cloth. While simpler than full-scale physics simulation, it provides physically plausible results in a fraction of the time, allowing interactive deformation editing. Moreover, thanks to the user driven material properties controls, our approach allows the user full control of the resulting models, while maintaining the physical plausibility.

In our setting, the material properties are expressed via stiffness coefficients, defined for the faces and edges of the mesh, which describe the bending and shearing flexibility of the surface. By adjusting these coefficients, we provide fine continuous control over the resulting deformations. Through rotation angle separation, our bending mechanism supports anisotropic bending stiffness, further enhancing deformation realism. In our setting these material properties can be user-driven using a simple paint-like interface to define them. Alternatively the properties can be data-driven, namely learned from a sequence of sample poses. Using the learning process we obtain physically correct material properties from physical simulation or manual character animation. By combining the data-driven and user-driven mechanisms, our system supports controlled realism, where the user can further refine and modify material properties derived via the learning process.

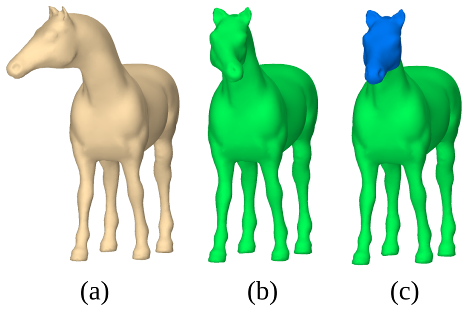
Figure 1. Turning the horse head. (a) original model. (b) using a uniform material. (c) using a non-uniform material.

1. Introduction

Mesh deformation is one of the most important tasks in modeling and animation of digital models. There are numerous applications for deformations in the computer animation and video game industries as well as in medical visualization and simulation systems.

Purely geometric deformation techniques generally rely solely on control points to describe a deformation, inde-

[*]e-mail: stpopa@cs.ubc.ca
[†]e-mail: djulius@cs.ubc.ca
[‡]e-mail: sheffa@cs.ubc.ca

pendent of the semantics of the underlying material. Thus, they usually require elaborate work to achieve kinematically sensible deformations. In contrast, physically based methods provide physically accurate behavior, but they are often computationally intensive and lack intuitive user-interfaces. For articulated models, the most established approach is to use a skeleton in order to simplify the task of defining deformation behaviour. However, skeleton contruction is a non-trivial task and skeleton deformations do not extend well to non-articulated models such as cloth.

In this paper we introduce a new concept of material aware deformations. We use material properties to provide fine continuous control of the surface behavior resulting in a unified framework for semantic deformation of articulated and non-articulated models.

Using a simple brush-like painting interface users are able to define bending and shearing stiffness coefficients across the mesh in order to control the deformation. For articulated models the user "paints" joint regions flexible achieving deformation behaviour equivalent to a skeleton (see fig hand). For non-articulated models the user can take advantage of the continuous range of the materials and specify gradient coefficients (see fig octopus) or a random looking pattern (see fig SMI + perlin noise)

Our method also supports anisotropic deformations allowing the user to specify different bending stiffness coefficients for arbitrary axis of rotation, thus providing finer control of the deformation. We are the first, to our knowledge, to support this kind of behavior.

In some cases the user may already have a set of sample deformations for a given model. We support a data-driven approach where our system can learn the material properties implicitly defined in the sample set. Users can then use these to create new poses consistent with the original ones without requiring manual definition of the material properties. Even when using the data driven coefficients, it is easy for the user to refine the aquired properties for finer control over the deformation resulting in a user-driven/data-driven hybrid system.

The rest of this paper is organized as follows: Section 2 reviews previous work on editing and deformation techniques. Sections 3 and 4 describe our deformation algorithm. Section 5 explains how we extend the method to support anisotropic bending stiffness properties. Section 6 describes how material properties may be learned by example. Section 7 shows some example results. Finally, Section 8 summarizes our work.

2 Previous work

Researchers have addressed the problems of mesh editing and deformation for over twenty years creating an impressive body of literature and generating several distinct frameworks to approach the same problem. One of the first, but still actively researched mesh deformation frameworks is that of space warping deformations [5, 16, 4, 6]. Since space deformations techniques transform the underlying space, rather than the vertices themselves, it is not possible to incorporate model specific properties such as materials into these techniques.

Another class of deformation methods that does incorporate material properties is physical simulation methods [24, 23]. Physical simulation based techniques provide physically accurate behavior, but they are often computationally intensive and lack intuitive user-interfaces.

For articulated models, a common approach is to use a skeleton in order to simplify the task of defining the deformation [7]. Using a relatively small number of joint angles the model can be posed easily, vertices are then positioned automatically according to their matching bones. A major drawback of skeletons is that they provide only two levels of stiffness, thus limiting type of deformations created. Our method provides continuous control over the stiffness of the mesh providing finer control of the deformation. Another challenge when modeling with skeletons is the task of creating the skeleton, and assigning the vertices to the appropriate bones. This is often done manually and it is very tedious and time consuming operation. To avoid artifacts at joins, vertex skinning methods [13] blend the effect of adjacent bones on each vertex. Our approach solves some of these issues. By replacing the skeleton with a material map we no longer have to worry about the skinning, and, by using the painting interface, the material map is arguably easier to construct than the skeleton.

Lately many techniques that operate directly on the mesh have been developed for editing and deformation [1, 20, 14, 17, 25, 11, 27, 15, 26]. While these methods are all efficient to compute and produce high quality results, they do not consider the material properties of the model, thus yielding rubber looking deformations. Yu et al. [25] perform 3D mesh deformations by means of gradient manipulation. First, users manually modify the position of some anchor vertices. Next, the resulting local triangle transformations are propagated to the rest of the mesh according to geodesic distances. Finally, vertex positions are found using the Poisson equation. It is shown by Zayer et al. [26] that propagation of the transformations according to geodesic distance is not optimal, and the authors suggest using harmonic fields as an alternative. Igarashi and Moskovitc [11] use a similar method for 2D meshes in which they manipulate the triangles of the mesh rather than the vertices in order to animate the mesh. This technique was first introduced by Alexa [3] to morph between 2D models. They show early research results for using stiffness coefficients to control the deformation; however, they acknowledge that extension to 3D may be difficult. Our propagation method is conceptually sim-

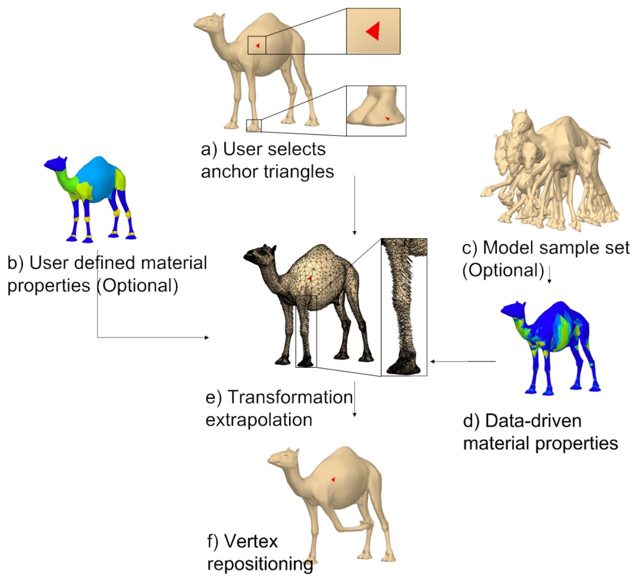

Figure 2. Algorithm flow

ilar to these methods. The most notable difference is that we incorporate isotropic and anisotropic material properties into our method to provide more control of the deformations. These properties are also used to create meaningful deformations based on a set of sample poses. Furthermore, our formulation supports a more flexible user-interface in that all anchor triangles have an active role in the deformation, thus allowing us to combine multiple transformations and not necessary only rotations.

Recently James and Twigg [12] and Sumner et al. [22] introduced methods for creating deformations based on a reference model and sample set of deformations. James and Twigg [12] automatically deduce a skeleton from a sample set of deformed models. Using the estimated skeleton and estimated blending weights they are able to create new deformations consistent with the sample set. Sumner et al. [22] use the set of sample models to create feature vectors that span the space of meaningful deformations. New poses are then generated within this space. Using our method, we are also able to use a set of sample poses as a source for automatically learning the stiffness coefficients of the mesh. These are later used to create new poses consistent with the samples. Moreover, using our framework it is very easy for artists to refine the learned weights to their satisfaction, thus providing a more controlled environment.

3 Method overview

We present a a two step method for 3D mesh manipulation that takes into account the intrinsic material properties of the model. After defining the material properties users manually select a small set of triangles, called *anchor* triangles (Figure 2(a)) and apply the desired transformation using a click and drag motion. We then calculate transformation for the remaining triangles of the mesh based on the anchor transformations and the material properties. We now proceed to describe the material properties.

Material properties — We use material properties to define the desired surface behavior under deformation. We distinguish between two types of behavior — bending and shearing. By defining non-uniform stiffness coefficients across the mesh with a paintbrush-like tool (Figure 2(b)), users are able to control these behaviors determining where each of them are to occur.

In addition to using non-uniform material properties, we also allow users to define anisotropic material properties to control the mesh bending. It is not uncommon for materials to bend differently with respect to different directions for instance, articulated models usually have joints that rotate around only a single axis.

We are also able to use a data-driven approach by automatically learning the material properties based on a set of example deformations (Figure 2(c)). By examining the shape of each triangle and the relative transformations between adjacent triangles within the entire sample set, we are able to identify degrees of stiffness and flexibility across the mesh. We then use this knowledge to assign appropriate stiffness coefficients to each triangle, thus ensuring that future deformations are consistent with the sample set.

Next, we explain how these stiffness coefficients, together with the user defined transformations at anchor triangles, are applied in our algorithm.

Transformation extrapolation — The first step of the algorithm is to automatically propagate the transformations from the anchors to the remaining triangles in the mesh. Finding optimal transformations for each triangle is non-trivial since these must comply with a number of constraints: First, the transformations must be continuous across the surface, thus resulting in a smooth looking deformation. Next, the transformations must also be as-rigid-as-possible, which is crucial in order to maintain the details of the original surface ([25]). Finally, in our setting we add a new additional constraint — the transformations must be consistent with the material properties previously defined. This final requirement is what ensures our deformations behave as desired.

We suggest that each triangle transformation be a weighted sum, or blend, of anchor transformations. Thus, our challenge is to find appropriate weights for blending that comply with the previous requirements. We formulate this as a linear optimization problem where the variables are the blending weights. We use the stiffness coefficients to ensure that the transformations in stiffer areas of the mesh are more rigid than those in more flexible ones. Since the solution only depends on the selection of anchors, we need only solve the system once. To perform the actual blending we use the matrix algebra defined by Alexa in [2]. We are aware of the discussions on the correctness and optimality of this method, thus we provide in Appendix A a brief discussion about this choice.

Vertex repositioning — It is easy to see that applying the resulting transformations to each of the triangles in the mesh will break the connectivity. Since adjacent triangles are not necessarily assigned identical transformations, the deformed common vertices of two adjacent triangles would not have the same position with respect to each of the transformations (Figure 2(e)). Therefore we apply a second step in which we find optimal positions for each of the vertices. We use a variation of [21] to find these positions such that each triangle is transformed as close as possible (in the least squares sense) to the previously calculated transformation (Figure 2(f)). We apply the shearing stiffness coefficients to the linear system to ensure that most of the distortion is concentrated in the flexible areas of the mesh.

Figure 2 summarizes our algorithm, and the following three sections describe it in detail: Section 4 explains how transformations are propagated and then optimal vertex positions are found, Section 5 explains how the method is extended to support anisotropic stiffness properties, and Section 6 explains how stiffness coefficients are estimated from sample deformations.

4 Method details

We begin this section describing how our material properties are defined (Section 4.1). Next, we define the gradient transformations and explain how these are propagated from anchor triangles (Section 4.2). Finally, we explain how optimal vertex positions are found (Section 4.3).

4.1 Material properties

We formulate our material properties in terms of face and edge stiffness coefficients, which are used to define the local degree of rigidity. This approach, as apposed to using skeletons to define rigid areas (bones) and flexible ones (joints) allows a much higher degree of control, including smooth variations of stiffness across the mesh.

We identify two measures that characterize surface behavior under deformation: bending and shearing. These correspond to the two types of coefficients we define:

- edge coefficients φ_{ij} that control the bending and are used to propagate the anchor transformations (eq. 1)

- face coefficients ψ_i that control the shearing and are used to find optimal vertex positions (eq. 3).

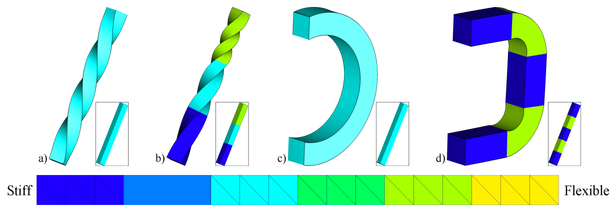

Figure 3. Twisting and bending a bar using two anchor triangles: (a), (c) uniform materials; (b), (d) non-uniform materials.

The bending resistance often depends on the rotation axis. For example, articulated models or physical materials exhibiting anisotropic deformation behavior. Therefore, we support anisotropic materials by specifying multiple edge bending coefficients for different axes of rotations.

We support both user-driven and data-driven methods for defining the material properties. In the user-driven setting we provide a simple paintbrush like tool to define the different degrees of stiffness. The users can paint both face and edge coefficients, or to simplify the interface, we can define the edge bending coefficients as the average of the adjacent face coefficients. We found that this typically works well, since there is usually a high correlation between the bending and shearing coefficients. Figure 3 demonstrates painting areas of different levels of stiffness on a 3D bar and the resulting deformations. For the data-driven approach, we estimate the material properties from a sample set of deformations automatically. Additionally, our formulation naturally supports user-intervention after the data-driven material estimation step. This is important in a production setting where animators require simple user controls to fine-tune the automatically generated results.

4.2 Transformation extrapolation

After the material properties and anchor transformations are defined we are ready to create the deformation. The entire mesh deformation can be expressed as a set of an affine transformations $(Ax+b)$ of local coordinate frames defined per triangle. The deformation gradient of each transformation is the matrix A, which encapsulates the triangle transformation up to the translational component. The three vertices of a triangle do not determine a local frame, thus we augment the three vertex positions with a fourth point found by offsetting one the vertices by the unit triangle normal ([21]). Labeling the vertices of a given triangle as v_1, v_2 and v_3 and the additional vertex v_4, the three vectors that define the local coordinate frame are: $(v_4-v_1, v_4-v_2, v_4-v_3)$.

After the user specifies the transformations for each of the anchor triangles we perform the first step of our algorithm. Deformation gradients defined at the anchor triangles are propagated o the remaining triangles of the mesh. We achieve this by using a weighted blending of anchor transformations. As previously noted, our challenge is to find appropriate weights for this blending subject to the material properties such that triangles in stiffer areas are assigned more similar and rigid transformations. We formulate this as a linear optimization problem:

$$\min_{\omega_i \in R^k} \sum_{(i,j)\in E} \varphi_{ij} \parallel \omega_i - \omega_j \parallel_2^2, \tag{1}$$

where the unknowns are $\omega_i \in R^k$ – the blending weights for face i. Each ω_i is a vector $(\omega_i^1, \omega_i^2, \ldots, \omega_i^k)$ where ω_i^j denotes the relative influence of anchor transformation j. k is the number of anchor triangles, E is the set of edges (excluding edges shared by two anchor triangles and boundary edges) and φ_{ij} are the bending stiffness coefficients of each edge.

When φ_{ij} is large, ω_i and ω_j will have similar values; thus, the resulting transformations of the two adjacent triangles i and j will also be similar, and the mesh may be considered as locally stiff. Similarly the converse argument can be made for small φ_{ij}.

Note that in this formulation, the weights ω_i depend only on the connectivity of the mesh and selection of anchor triangles; therefore, the weights need to be computed only once per selection of anchors for each deformation. Also, note that our weights are barycentric coordinates with respect to the anchors. This is an important difference compared to the method of Yu et al. [25] who based their weights on geodesic distances. As a result our method does not suffer from the propagation problems noted by Zayer et al. [26].

In order to perform the actual blending we use the commutative matrix algebra defined in [2]. By defining two new operations denoted by $\oplus$ and $\odot$, the blending transformations T_j with weights ω^j are formulated as:

$$T_i = \bigoplus_{j=1}^{k} \omega_i^j \odot T_j. \tag{2}$$

Further details on these operators are found in Appendix A. Figure 2(e) illustrates the propagated transformations between two anchors on the camel leg.

4.3 Vertex repositioning

Since, adjacent triangles do not typically have identical transformations, applying each of the transformations as-is would result in ambiguous positions for the two shared vertices. Therefore, a second stage to compute the optimal position for each vertex is required.

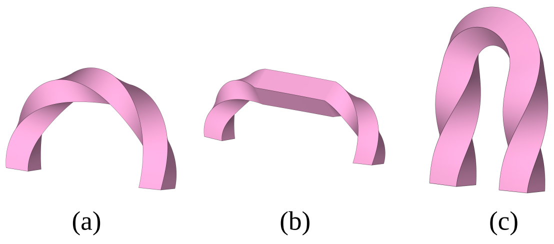

Figure 4. Bending and twisting a bar with anisotropic materials. (a) uniform material; (b) center region marked as isotropically stiff; (c) center modified anisotropically to allow bending but not twisting.

Optimal vertex positions are found such that the gradient transformation of each triangle remains as close as possible (in the least squares sense) to the previously calculated transformation [21]. In order to take material properties into account, we use the face stiffness coefficients to direct most of the distortion according to the flexibility of the mesh.

Sumner and Popovic [21] show that the transformation gradients can be expressed in terms of the vertex positions before and after the deformation:

$$A = \tilde{V}V^{-1},$$

where

$$\tilde{V} = (\tilde{v}_4 - \tilde{v}_1, \tilde{v}_4 - \tilde{v}_2, \tilde{v}_4 - \tilde{v}_3),$$
$$V = (v_4 - v_1, v_4 - v_2, v_4 - v_3),$$

and $\tilde{v}_i$ is position of vertex v_i after applying the deformation transformation.

In our setting we must also account for the shearing stiffness properties when solving linear optimization problem. Thus, we added ψ_i to the formulation which is as follows:

$$\min_{\tilde{v}} \sum_{i}^{n-k} \psi_i \|\tilde{V}_i V_i^{-1} - T_i\|_F^2, \tag{3}$$

where the unknowns are the new vertex positions $\tilde{v}$, V_i and $\tilde{V}_i$ are the local frames before and after applying the deformation and T_i are the previously calculated transformations. As before, large ψ_i will result in transformations very close to the originally computed T_i which are rigid. Thus, distortion will be distributed in the mesh according to the shearing coefficients.

This is reformulated as a linear optimization problem:

$$\min_{\tilde{v}} \|\Psi(A\tilde{v} - t)\|_2^2, \tag{4}$$

where A is a sparse matrix constructed using the pre-deformation local frames V, t is a vector composed of all the elements in T_i and Ψ is a diagonal matrix composed of ψ_i.

The solution of this system as defined above is a new position for each of the vertices up to a global translation of the model. To anchor the model in place, we fix the position of a single vertex by removing the corresponding variable and pre-multiplying its known position with the appropriate elements of A into the vector T ([22]). In fact, multiple vertices may be set at fixed positions to apply boundary constraints such as regions of influence.

5 Anisotropic materials

The formulation presented above provides a single scalar per edge bending coefficient. This assumes that bending flexibility is a non-directional property, which need not be true in practice. In this section we extend our method to allow multiple bending stiffness coefficients depending on the axis of rotation. This allows us to provide even finer control over deformation behavior using the same framework.

Instead of using one bending stiffness coefficient per edge we define three different coefficients, corresponding to rotations around three orthogonal axes x, y and z. Using these coefficients we solve eq. 1 three times to get three weight vectors ω_i^x, ω_i^y and ω_i^z. In the isotropic case, we used ω_i as weights for blending the anchor transformations. In the anisotropic case we need to decompose the anchor transformations into rotations around the three axes x, y and z in order to apply the newly acquired weights.

As noted in [2], three matrices R_x^θ, R_y^θ, R_z^θ denoting rotations by an angle $\theta \in [0,\pi]$ around three orthogonal axes x, y, and z form a basis of the sub-space of rotations. Thus, for any given rotation matrix T we can find a commutative decomposition such that

$$T = a \odot R_x^\theta \oplus b \odot R_y^\theta \oplus c \odot R_z^\theta .$$

Furthermore, the coefficients a, b, c are found by simply computing the following inner products:

$$a = < log(T), log(R_x^\theta) >,$$

where

$$< A, B > = \sum A_{ij} B_{ij}.$$

b and c are found in a similar manner.

By defining $X_j = a_j \odot R_x^\theta$, $Y_j = b_j \odot R_y^\theta$ and $Z_j = c \odot R_z^\theta$ we get a unique commutative decomposition of any transformation T_j into three rotation matrices X_j, Y_j and Z_j around the three orthogonal axis of rotations x, y and z.

We now can find the local triangle transformations T_i by rewriting eq. 2:

$$T_i = \bigoplus_{j=1}^{k} \omega_{ij}^x X_j \oplus \omega_{ij}^y Y_j \oplus \omega_{ij}^z Z_j ,$$

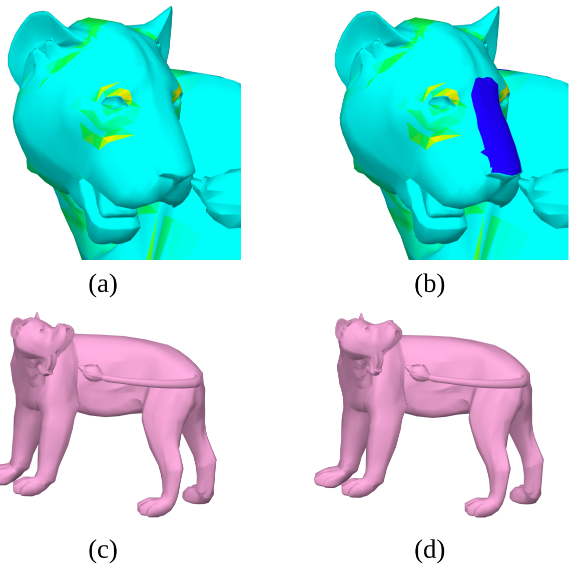

Figure 6. Refined materials: (a) estimated materials (Color coding is rescaled for comparison to the refined version). (b) modified pose created by rotating the lower jaw while using only the estimated materials; (c) manually refined materials; nose is marked as stiff (d) modified pose using the refined materials.

where T_i are the unknown transformations, $\omega_{ij}^x, \omega_{ij}^y$ and ω_{ij}^z are the three blending weights and X_j, Y_j and Z_j are decompositions of the anchor triangle transformations T_j such that $T_j = X_j \oplus Y_j \oplus Z_j$.

Since we only define anisotropic coefficients for bending the rest of the algorithm continues as previously described. Figure 4 shows an example of using anisotropic coefficients while bending a 3D bar. It is easy to see how each flexible area responds differently depending on the axis of rotation.

6 Material learning

It is not uncommon to have sets of sample deformations for a single model. In this case we would like to be able to create new, meaningful, deformations which are consistent with the sample set. The following section describes our method of acquiring the material properties from sample deformations automatically.

Given a reference mesh P_0 with n vertices and m triangles and a set of deformed meshes with the same connectivity P_i $i = 1 \ldots k$, we first compute the deformation gradient for each of the triangles $j = 1 \ldots m$ in P_i as follows:

$$T_{ij} = \tilde{V}_{ij} V_{0j}^{-1} \quad i = 1 \ldots k, j = 1 \ldots m,$$

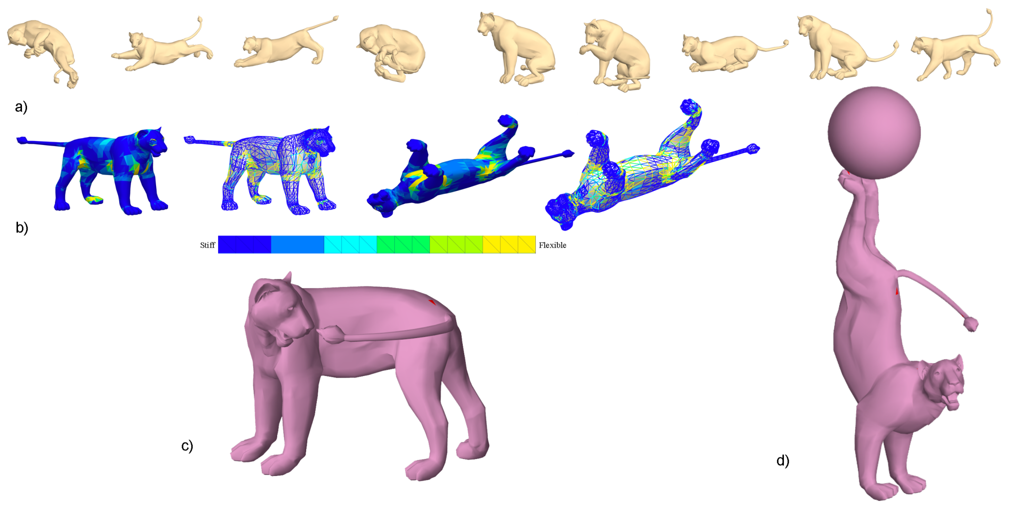

Figure 5. Estimation of stiffness coefficients from sample poses. (a) sample poses; (b) estimated bending and shearing coefficients on edges and faces respectively; (c) tail chase pose created by bending the head toward the tail with two anchor triangles on the back; (d) handstand created by rotations applied to the back legs, head, and tail.

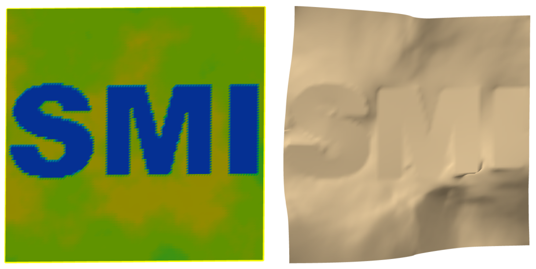

Figure 7. A wrinkled sheet of paper with a texture map as the source for the material properties.

where V and $\tilde{V}$ are the local frames in the reference and deformed meshes.

Next, we decompose each deformation gradient matrix using the polar decomposition $T_{ij} = S_{ij}Q_{ij}$ s.t. Q_{ij} is a rotation transformation, and S_{ij} is a combination of scaling and shearing ([19, 12, 22]).

We observe that rigid pieces of the model rotate as single units, therefore adjacent triangles in a rigid part will have similar Q matrices. In contrast, flexible regions might exhibit large differences between the rotations of adjacent triangles. Thus, we estimate the bending coefficients based on these differences. Furthermore triangles in flexible regions are the ones to deform more due to shearing, thus $||S||_F^2$ for triangles in those regions is larger and is used to estimate the shearing coefficients.

The bending coefficient φ_{ij} of each edge is estimated based on the transformations of two triangles T_i and T_j sharing the edge. In each example the difference between the two matrices Q_i and Q_j of adjacent triangles is equivalent to the amount of bending around the shared edge. Since each sample only exhibits bending around a subset of the edges we need to combine values from multiple samples. The maximum difference observed for each edge is equivalent to the maximum bending around the edge among all samples and is therefore used to define the bending coefficient. The values are then scaled to be in $[\varepsilon, 1]$.

$$\tilde{\varphi}_{ij} = \max_{l=1\ldots k} ||Q_{li} - Q_{lj}||_F^2 \quad (i,j) \in E,$$

$$\varphi_{ij} = 1 - \frac{\tilde{\varphi}_{ij}}{\max_{l,m\in E} \tilde{\varphi}_{lm} + \acute{\varepsilon}} \quad (i,j) \in E.$$

For each face j we estimate a shearing coefficient ψ_j based on the maximum distortion of the face found in the set of example deformations. The values are then scaled to be in $[\varepsilon, 1]$.

$$\tilde{\psi}_j = \max_{i=1\ldots k} ||S_{ij} - I||_F^2 \quad j = 1\ldots m,$$

$$\psi_j = 1 - \frac{\tilde{\psi}_j}{\max_{l=1\ldots m} \tilde{\psi}_l + \acute{\varepsilon}} \quad j = 1\ldots m.$$

These ψ_j are later used as the diagonal elements of Ψ in eq. 3 to direct the distortion. We found that clamping the top 1% values before scaling greatly improved our results.

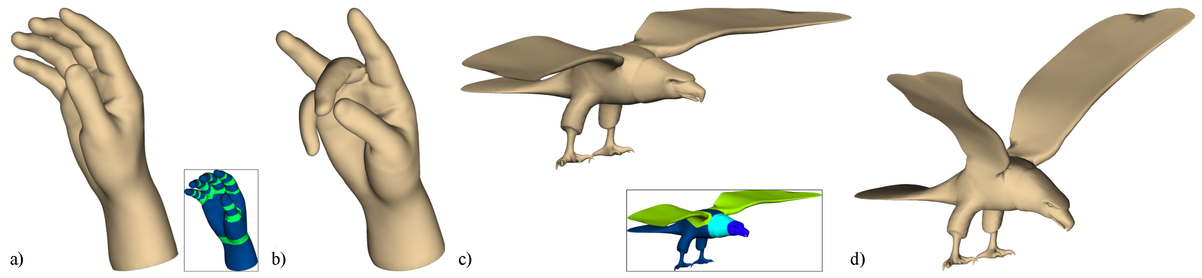

Figure 8. Deformation results: (a), (c) original models; (b), (d) deformation using our method.

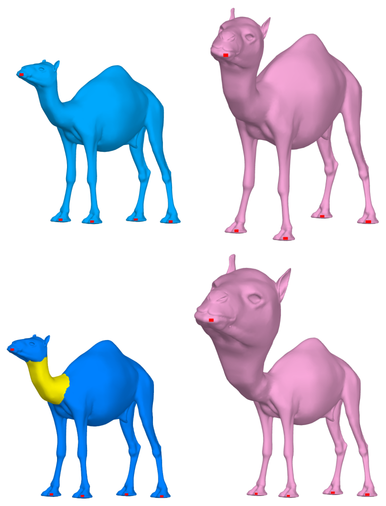

Figure 9. Camel head scaling. Top: uniform materials. Bottom: non-uniform materials.

Figure 5 shows an example of estimated bending and shearing coefficients from a sample sequence of deformations. We use these as a basis for creating new poses which were not in the sample set.

Our method is a much simpler method than those of James and Twigg [12] and Sumner et al. [22], who also use polar decomposition as a first step of their methods. Nevertheless, our experimental results are credible. Furthermore, our method exhibits two nice properties: our learning algorithm is linear in the number of sample poses and the learned weights can be further refined by the user for finer control.

7 Results

To test our method we used the Graphite [9] framework to create an interactive deformation tool. Material properties are defined using a simple color map with a paintbrush interface. To deform models users mark anchor triangles and then apply rotations and scaling to them by dragging the mouse. Once the anchors are defined, we used a SuperLU [8] solver to compute the solutions of the equations 1 and 4. Since in both systems the coefficient matrix depends only on connectivity and the vertex position of the undeformed mesh respectively, we precompute the inverse of the coefficient matrix, allowing the system to output interactive visual feed-back to the user.

We also support the concept of deforming only regions of interest confining all the calculations to triangles within that region. When using such a region of interest, the vertices on its boundary are constrained to remain in their initial position.

Our results demonstrate that the method may be applied to a large variety of model types. Figures 1, 2(f), and 8 demonstrate how a simple coloring scheme for defining the material properties allows us to easily deform articulated models. In the hand example from figure 8 , we painted the joints with a flexible material. Then, to avoid artifacts on transition lines between materials, we smoothed the stiffness coefficients. For this example, we used five anchors for each finger, but we deformed them independently using a region of influence for each finger. The eagle deformation in figure 8 uses simple coloring of the head, neck and wings to illustrate that the head is the most rigid part while the wings are the most flexible. The deformation was created using one anchor on the tip of the beak and another four anchors on the back between the wings. Next, the wings were raised using two additional anchor triangles.

A more complex stiffness scheme is required when modeling some non-articulated models that exhibit inherently different flexibility along the surface. Tree branches, plant stems, and octopus tentacles are examples of models where the flexibility is proportional to the girth of the model. Figure 10 demonstrates how non-uniform material properties allow us to easily define correct bending behavior for the tentacle. When using a uniform material the tentacle takes the unnatural shape of an arc; however, using non-uniform materials results in the more natural shape of a spiral with the tip bending more than the base. In both deformations we applied the same rotation transformation to an anchor

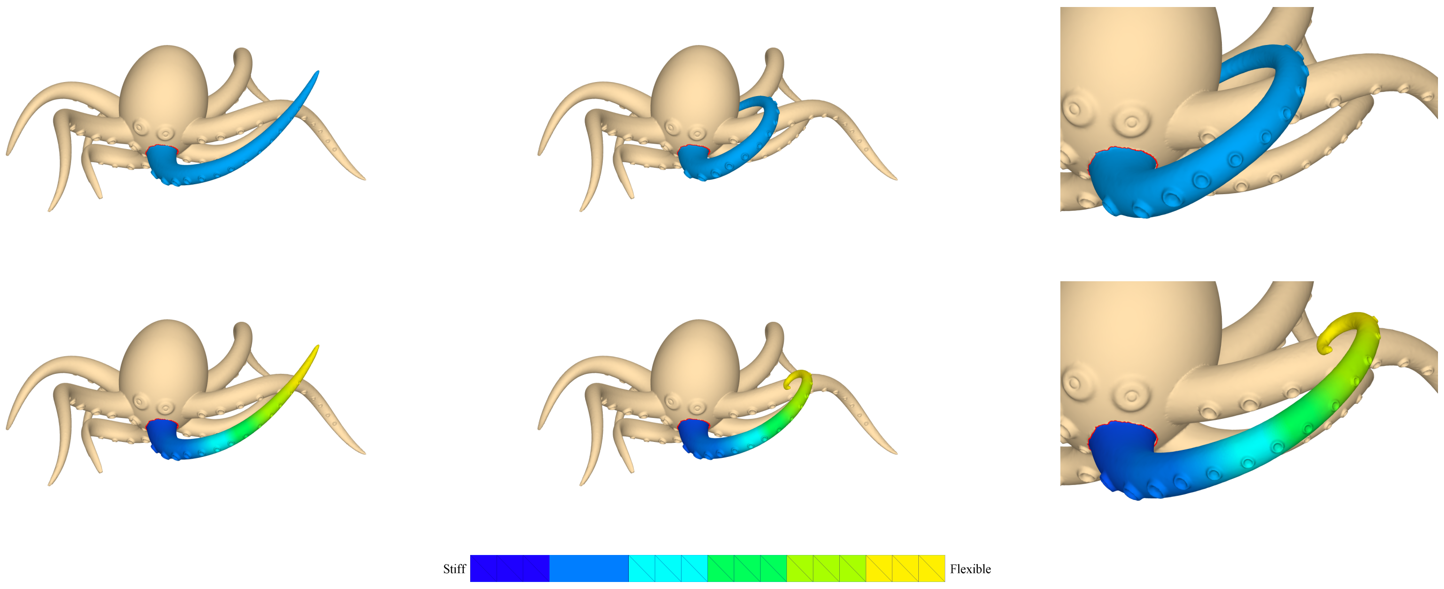

Figure 10. An octopus tentacle deformed under different materials. We defined only the amount of rotation applied at the tip of the tentacle, while the final position of the tip is calculated automatically. Top: uniform stiffness materials; bottom: smooth variation of stiffness resulting in a spiral like curve.

triangle at the tip of the tentacle, while the final position of the tip was computed automatically.

Figure 7 demonstrates an additional advantage of using our method compared to skeleton based deformation. In order to deform the sheet of paper we used a texture map to define the material properties. The bold letters define very stiff coefficients, while the rest of the model varies in degrees of flexibility. It is not possible to achieve such an effect using skeletons.

We have also tested our method for estimating the stiffness coefficients from sample poses. Figures 2(d) and 5 and demonstrate our results. For sample poses we used models obtained from [21]. In both cases the estimated stiffness coefficients are plausible, showing more flexible regions at the joints and stiffer regions along the bones as expected. Figures 5(c) and (d) show examples of new anatomically plausible poses created using the estimated coefficients and only eight anchor triangles located at the head, tail, feet, and back. One limitation of estimation from samples is the fact that we are only able to detect flexible regions of the model if at least one of the poses exhibits some bending or shearing in those regions. In Figure 6 we show how we can easily refine the estimated coefficients to create new poses which are not spanned by the example pose set. For comparison we also show the resulting deformation by performing the exact same anchor transformation on the lower jaw without refining the stiffness coefficients.

Finally, Figure 9 demonstrates a combination of scaling and rotation applied to the head of the camel. The deformation was created using four fixed anchors on the bottom of the feet and one anchor on the tip of the nose where the single transformation was applied. By applying different materials we are able to easily control the distribution of the deformation across the mesh. In fig 9(bottom), even though the anchors are located very far apart on the mesh (i.e. head and feet) the rigid parts (i.e. head and body) are well preserved and all the variation in scale happens only at the neck.

The leitmotif of our work is that factoring the surface behavior in a preprocessing stage greatly simplifies the deformation operations in terms of numbers of control primitives without compromising quality or control. We showed that our technique can be applied to a wide range of models producing complex results with only few anchors.

Table 1 summarizes the statistics of the various models used in our results. We ran our system on a 3GHz Intel Pentium IV with 2Gb of RAM.

8 Summary and future work

We presented a new mesh deformation method centered around material properties. Material properties, as defined here, provide a simple mechanism that allows full control over the behavior of the surface under deformation. Our

	Bar	Hand*	Lion	Cloth	Horse	Eagle	Octopus*	Camel
#Faces	1596	6274	9996	19602	19996	29232	36542	43775
#Anchors	2	3	8	10	2	7	5	5
Material learning:								
#Poses			9					10
Estimation (s)			2.1					9.5
Preprocessing:								
Weight calculation(s)	0.02	0.29	1	4.86	0.86	2.58	3.51	3.53
Vertex repositioning factorization(s)	0.02	1.38	1.24	4.43	5.58	5.82	9.58	10.13
Total (s)	0.04	1.67	2.24	9.29	6.44	8.40	13.09	13.66
Real-time:								
Blending (ms)	1	2	3	7	6	10	12	15
Repositioning (ms)	2	2	3	8	9	13	18	20
Total (ms)	3	4	6	15	15	23	30	35

Table 1. Model deformation timings. For the octopus and hand models we restricted the region of influence to one tentacle and one finger respectively.

method provides a simpler and more powerful alternative to skeletons, allowing various degrees of stiffness. Furthermore, it is the first method to our knowledge, to support anisotropic material behavior.

Material properties can be user-driven, where the stiffness coefficients are specified using a simple brush-like interface, data-driven where stiffness coefficients are deduced from a set of given poses, or a combination of the two where the user can override data-properties of the material.

The formulation is simple and efficient, requires solving only two linear systems, and thus works at interactive rates. The resulting deformations are as-rigid-as-possible subject to the material properties and therefore maintain the shape details as seen in our results.

For future research we would like to improve the anisotropic model presented to support property definition with respect to local coordinate frames. While this should not pose any problem in the reconstruction phase, it is challenging to construct blending weights ω_{ij} which are consistent across different coordinate frames.

9 Acknowledgments

To appear in final version

References

[1] M. Alexa. Local control for mesh morphing. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 209, Washington, DC, USA, 2001. IEEE Computer Society.

[2] M. Alexa. Linear combination of transformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387, New York, NY, USA, 2002. ACM Press.

[3] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In K. Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 157–164. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[4] A. Angelidis, M.-P. Cani, G. Wyvill, and S. A. King. Swirling-sweepers: Constant-volume modeling. In *Pacific Conference on Computer Graphics and Applications*, pages 10–15, 2004.

[5] A. H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1984. ACM Press.

[6] M. Botsch and L. Kobbelt. Real-time shape editing using radial basis functions. In *Computer Graphics Forum, (Eurographics 2005 proceedings)*, volume 24, pages 611–621, 2005.

[7] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic;. Interactive skeleton-driven dynamic deformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 586–593, New York, NY, USA, 2002. ACM Press.

[8] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[9] Graphite, 2003. http://www.loria.fr/∼levy/Graphite/index.html.

[10] F. S. Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, 1998.

[11] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[12] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, 2005.

[13] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[14] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[15] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005*, page accepted for publication. ACM Press, 2005.

[16] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, New York, NY, USA, 1986. ACM Press.

[17] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT*, pages 68–75, 2004.

[18] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press.

[19] K. Shoemake and T. Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[20] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM Press.

[21] R. W. Sumner and J. Popovic. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.

[22] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popovic;. Mesh-based inverse kinematics. *ACM Trans. Graph.*, 24(3):488–495, 2005.

[23] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278, New York, NY, USA, 1988. ACM Press.

[24] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM Press.

[25] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[26] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.

[27] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. volume 24, pages 496–503, New York, NY, USA, 2005. ACM Press.

A Matrix algebra

Geometric transformations are typically represented as square matrices. Matrix multiplication is used to compose and apply the transformations. There are a few key shortcomings of this representation:

- Rotation transformations cannot be interpolated by interpolating the matrix elements.

- Matrix multiplication is not commutative

Both these properties are crutial for us in order to propagate and later decompose anchor triangles.

To deal with these issues a number of interpolation methods for rotations have been developed over the years. When dealing with rotations there are three desired properties: torque-minimization, constant speed, and commutativity. Current quaternion interpolation methods exist, altough none exhibit all three properties. SLERP [18] exhibits constant speed and minimal-torque. LERP, popularized by Casey Muratori, is both commutative and of minimal-torque. The the exponential map interpolation [10] is both commutative and of constant speed.

More recently a commutative algebra of (almost) general transformations that supports matrix blending and interpolation was introduced by Alexa in [2]. This algebra is commutative and interpolates transformations with constant speed. Furthermore, it is not limited to rotations only, thus simplifying the task of dealing with combinations of rotations and scales.

We have chosen to use the later method since it answers both our requirments. We now give a brief overview of the blending operators defined in this algebra.

By defining two new operations denoted by $\oplus$ and by $\odot$ (corresponding to matrix addition and scalar multiplication), blending transformations T_i with weights ω_i becomes $\bigoplus \omega_i \odot T_i$.

The two operators are based on matrix *exp* and *log* operators defined as follows:

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!},$$

$$A = \log(X) \Leftrightarrow \exp(A) = X.$$

Alexa [2] shows that this sum is well defined and closed for 3x3 rotation matrices and non-uniform scales under some minimal conditions. The blending formula is defined as follows:

$$\bigoplus_{j=1}^{k} \omega_{ij} \odot T_j = \exp(\sum_{j=1}^{k} \omega_{ij} \log(T_j)).$$

More details as well as numerical methods to compute these operations are found in [2].