Computers & Graphics ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Computers & Graphics

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

Technical Section

Constrained illustrative volume deformation

Carlos D. Correa [a,*], Deborah Silver [b], Min Chen [c]

[a] University of California, Davis, USA
[b] Rutgers University, USA
[c] University of Wales, Swansea, UK

ARTICLE INFO

Keywords:
Illustrative deformation
Displacement mapping
Volume rendering
GPU-based rendering

ABSTRACT

Interactive volume deformation has received a lot of attention recently thanks to the advances of graphics processing units. The ability to transform volumetric objects constitutes an essential tool for generating illustrative visualizations. Previous approaches to this problem have adapted techniques used for surface meshes and applied it to an embedding mesh. This method, however, often results in low quality due to the resolution of the mesh. Other alternatives consider the volume as a homogeneous collection of points, and therefore, cannot simulate physical laws that govern the deformation of different materials. In this paper, we aim at obtaining high-quality illustrations of deformable volumes that mimic physically inspired constraints. For example, although skin and muscle can be illustrated as deforming elastically, bone tissue should move rigidly. Here, we show that we can obtain constrained illustrative deformations with algebraic operations on displacement maps. Through a number of examples, we show that this is a faster alternative to costly physical simulations, with wide applications in computer-aided medical illustration, interactive volume manipulation and data exploration.

1. Introduction

Illustrative deformation is a family of interactive techniques that deform a graphical model to obtain an illustrative visualization. For example, for medical illustration, one may interactively manipulate a volumetric model of a scanned human body to obtain illustration of typical surgical effects such as bending, peeling, and cutting open. While this family of techniques does not exclude physically based deformation, the goal is to achieve a realistic appearance of the deformed model, rather than physical accuracy of the deformation interaction, such as force feedback.

Truly physically accurate deformation is difficult to obtain for volume datasets because most physically based simulation relies on surface meshes. Since a volume is a sampled representation of a continuous field, simulating cuts are also difficult via mesh. Moreover, accurate physically based deformation would require a precise specification of material properties of every voxel, and embedding of a volume in a 3D mesh where physics is applied. Since a deformed volume has to be reconstructed from the deformed mesh using interpolation, the quality obtained is usually low, and depends largely on the tessellation resolution of the mesh. In addition, the speed for simulating a slightly complex volume model would render the computation non-interactive.

In recent years, illustrative volume deformation using empirical (i.e., non-physically based) displacement operations has shown to be a practical technique for creating illustrative visualization featuring various deformation effects (e.g., [1,2]). One limitation of the existing methods is that all voxels to be deformed response to the force in the same way. For instance, when the knee model in Fig. 1(a) is deformed using axis-aligned deformation [3], the bone will be deformed in the same way as soft tissues around it as shown in Fig. 1(b). Although one may use feature-aligned manipulation [1] to mask the bone as a rigid object, as shown in Fig. 1(c), the rigid bone does not influence the deformation of the soft tissues around it. It is thus desirable to make soft tissues around the bone deform non-uniformly under the influence of the rigid bone. This provided this work with the motivation.

In this work, we introduce the notion of *constrained displacement fields*, which allows us to confine the deformation within certain limits. This is useful for preventing self-intersections and avoiding collisions with features of interest. Furthermore, it allows us to define deformation on heterogeneous materials, where parts of an object move rigidly and others elastically. Fig. 1(d) illustrates the application of this method to the knee model in Fig. 1(a). In Fig. 1(d), we can observe that the bone does not response the deformation force as a rigid part, while the soft tissues around the bone deform not only according to the force, but also under the constraints of the bone. Our novel contribution is that we achieve this through an empirical deformation function that is linear combination of displacement fields. Without using

* Corresponding author.
E-mail address: correac@cs.ucdavis.edu (C.D. Correa).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.01.007

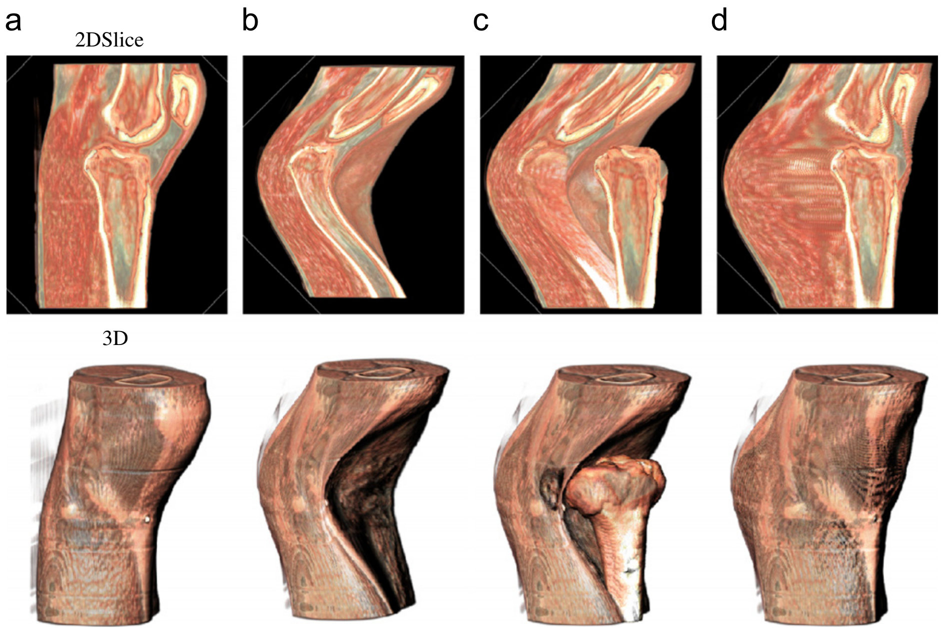


Fig. 1. Deformation of a knee CT scan. We compare different methods for deformation. Axis-aligned deformation transforms all voxels without regards on their material. We see that tissue and bone are applied the same transformation. Feature-aligned deformation [1] allows the bone to stay rigid while everything deforms around it. However, it results in self-intersections. Finally, constrained deformation ensures that no unwanted intersections are produced: (a) Original; (b) axis-aligned; (c) feature-aligned and (d) constrained.

physically based methods, we can maintain the interactive rate as unconstrained deformation [1], hence maintaining the usability of the tool for illustrative volume deformation. Previous attempts at introducing constraints can create compelling volume illustrations that do not only mimic the appearance of a reference illustration, but also the shape style [4]. Our approach is a different mechanism to control the effects of deformation to mimic the physical constraints of anatomical features.

Our approach is amenable to contemporary GPUs, as deformation is defined as a look-up on a displacement field, which can be efficiently stored as a 3D texture.

The rest of the paper is organized as follows: Section 2 describes the state-of-the-art in volume deformation, including the notion of discontinuous displacement mapping. To allow constrained deformation, we must be able to combine algebraically these displacements. To this purpose, Section 3 describes a series of unary and binary operations on displacement maps, including addition and modulation. Section 4 describes our method that uses algebraic operations on displacements to realize constrained deformation. We describe a series of applications in illustrative visualization, such as collision-free volume manipulation and fast simulation of rigidity constraints.

2. Background

Illustrative visualization: Illustrative visualization is mainly seen as a tool for the effective communication of knowledge. The goal of illustrative visualization is to develop software that enables domain experts and scientists to make illustrations of their work [5]. The main illustrative visualization technology falls into two classification. The first one relies on the manipulation of viewing attributes of the rendering engine and optical attributes of the objects, such as non-photorealistic rendering [6,7], ghosted views [8], magic lenses [9], ClearView [10] and close-ups [11]. The second one relies on volume deformation to effectively resolve the occlusion problem [1,12]. In [13], Correa et al. proposed a data exploration framework, which combines active manipulation of the spatial data with opacity and color transformations. In [14], Li et al. present a system for creating and viewing interactive exploded views of complex 3D models.

Volume deformation: The general notion of a *sampled object representation* (SOR) is a set of samples $V = \{(\mathbf{p}_i, v_i) | i = 1, 2, \ldots, n\}$, where v_i is a value of a specific data type (e.g., Boolean, scalar, vector or tensor), which represents some property at each sample location $\mathbf{p}_i$ in k-D Euclidean space E^k. Typically these samples are associated with a spatial domain Ω, which is normally continuous or consists of several disjoint sub-domains. An object specified by an SOR is thus a function $F(\mathbf{p})$ that defines the value at every $\mathbf{p} \in \Omega$ [15].

Deformation refers to the intended change of geometric shape of an object under the control of some external influence such as a force. We can classify deformation along two dimensions. Depending on the underlying model, we can refer to deformation methods as either physics-based or empirical. Most of the physics-based approaches have been devoted to mesh deformation. For an up-to-date compendium of such techniques, see Nealen et al.'s survey [16]. Models such as mass-springs and finite elements are used to ensure desired physical properties such as conservation of mass, elasticity and viscosity. Extending this to volumetric objects has been proposed in the form of tetrahedral meshes, or other embedding geometry. Because of the size of volumetric objects compared to meshes, the results obtained by this method are usually of low quality. Empirical approaches, in contrast, have little or no physics in their computation. Methods include free-form [17], global and local deformation [18], sketch-based [19] and skeleton driven deformation [20].

Another dimension for classifying deformation is determined by the stage at which it is applied. Most mesh-based deformations are done at the *modeling* stage, where each application of deformation leads to a new mesh, which can be rendered using traditional methods. Deformation at the *rendering* stage has recently surfaced thanks to the programmability of GPUs [21]. Early volumetric deformation approaches were applied at the modeling stage, [22–26]. However, this is impractical for interactive applications. One of the most notable problems is the need to re-sample at possibly higher rates, which would lead to huge data sets. For this reason, volume deformation has been usually tied to the rendering process. Examples of this type include ray deflectors [27], free-form lattices [28], pointwise deformation of pre-segmented volumes [2], splitting operations [29], spatial transfer functions [30], and chain-mail algorithms [31].

In this paper we focus on interactive non-physically based deformation. Unlike mesh-based models, deformation achieved by simply transforming the voxels is of limited use. Instead, analogous to images, volumes can be deformed via *inverse warping*. To understand this problem, we can define deformation as a transformation of points $T : \Omega \mapsto \Omega'$ from the deformed domain Ω to the original domain Ω'. As a general approach, we assume that this transformation can be defined as a displacement $D : R^3 \mapsto R^3$, such that

$$\mathbf{p}' = \mathbf{p} + D(\mathbf{p}) \tag{1}$$

for every point in the deformed domain Ω. We call this method *direct inverse warping*. Direct inverse warping has been applied before in the form of spatial transfer functions [30] and ray deflectors [27] and deformation shaders [28]. Unlike our approach, they defined the inverse mapping T as a general function, instead of a displacement. Although this is more general, we show that displacements can be generalized and operated algebraically, are easier to implement in the GPUs, and demand less computational complexity.

Previous approaches to interactive volume deformation used a hybrid approach, whereby a set of control points is deformed first using a forward transformation, and the final deformation is obtained via interpolation. We refer to such methods as *indirect space warping*. Fig. 2 shows the difference between the two inverse warping approaches. Methods in this category include free-form deformation of volumes [32], direct deformation of trilinear patches [33] and skeleton-based techniques [26,34,35].

One of the problems with indirect space warping is the reliance on a proxy mesh, as defined by the control points. Because the deformed object is obtained via interpolation, the smoothness and quality of the resulting image depends on the resolution of the proxy mesh.

The majority of previous work on volume deformation considers the transformation function as continuous. Particularly for surgery and medical illustration, deformations often contain discontinuities. This has been applied successfully for indirect space warping approaches [27,28], because of the ability to use a forward transformation at the control points. Similar to the continuous deformation problem, the quality of the cuts depends on the resolution at which the mesh is re-tessellated. See Fig. 3(a). For direct space warping, this is more complicated, as it implies that the function D is no longer continuous. Chen et al. [30] devised a solution to this problem with the introduction of a special invalid point $\varnothing$ to signify discontinuities. The deformations are computed along all points of the domain Ω, but the contribution to the resulting image is null for those points in the cut, as depicted in Fig. 3(b). We observed that this definition, although effective, is not easily implemented in the GPU when the deformation is defined as a displacement field, due to the intrinsic tri-linear interpolation of current GPU hardware. Instead, a better representation, which we call *discontinuous displacement*, is required.

Discontinuous displacement maps: In order to deform volumetric objects, the inverse mapping function must be continuous. This condition assumes that the volume is defined as a sampling of a continuous scalar field in some spatial domain. That is, let us define $f' : \Omega' \mapsto R$ a scalar field in the undeformed domain Ω'. There exists a corresponding scalar field f in the deformed space Ω, such that for every point $\mathbf{p} \in \Omega$

$$f(\mathbf{p}) = f'(\mathbf{p}') \tag{2}$$

where $\mathbf{p}' = \mathbf{p} + D(\mathbf{p})$. This condition, however, does not hold for a discontinuous deformation, such as a cut or an incision. There are two ways of dealing with this situation. (1) As a geometric transformation, which modifies D such that certain invalid points are considered out-of-bounds and therefore are handled as empty space, or (2) as an optical transformation, which modifies f such that certain points are considered as transparent and therefore automatically become empty space. Because the latter is more suitable for GPU implementation, we chose this approach, as follows. We define a mask field A, such that

$$f(\mathbf{p}) = \begin{cases} f'(\mathbf{p}') & A(\mathbf{p}) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We assume that, for the purposes of rendering, a scalar value of 0 is considered as transparent, and therefore, it is suitable for representing empty space. When analyzing the gradients of the resulting transformation, we can also estimate the normals of the deformed volume for lighting. The normal of a deformed point $\mathbf{p}$ can be estimated as

$$\overrightarrow{\mathbf{n}}^{(p)} = (\mathrm{I} + \mathrm{J}_D(\mathbf{p}))^{\top} \overrightarrow{\mathbf{n}}^{(p')}$$

where J_D is the Jacobian of the displacement D.

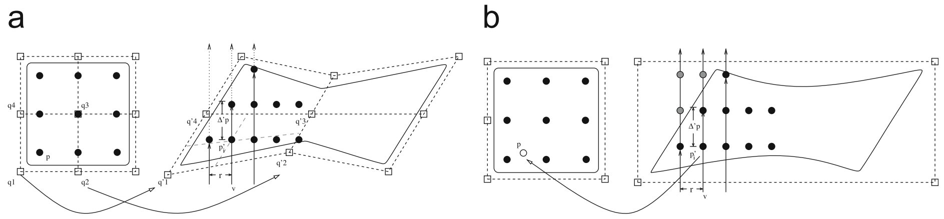


Fig. 2. Continuous deformation methods using inverse warping: (a) Indirect warping and (b) direct warping.

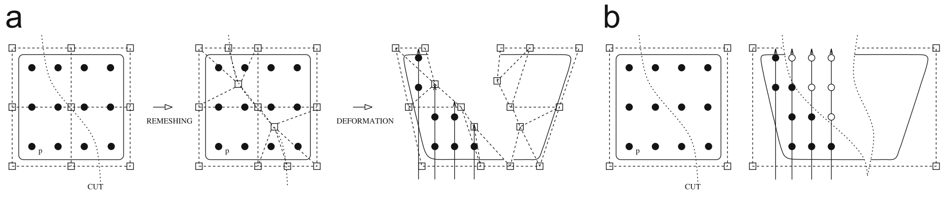


Fig. 3. Modeling cuts in inverse warping methods: (a) Cuts in indirect warping and (b) cuts in direct warping.

Therefore, we can formally define a discontinuous displacement map as a tuple D,A, where $D : \Omega_D \mapsto [0,1]^3$ is a displacement field, and $A : \Omega_D \mapsto [-1,1]$ is the *continuity mask* of that displacement field, for a given displacement domain Ω_D. Here, for simplicity, we assume that the displacement domain coincides with the volumetric domain.

Creation of displacement fields: In our approach, we follow the metaphor of templates to define our deformation. For example, there are templates for a knife cut, retractor cut, bending and twisting among others. These are created analytically by sampling a displacement function at discrete positions x,y,z. For example, a knife operator, that splits a volume in two along the X axis (with gap size 2σ), can be created as a displacement D and an alpha mask A

$$D \cdot x = \begin{cases} \sigma & x < 0.5-\sigma \\ -\sigma & x > 0.5+\sigma \\ 0.5-x & \text{otherwise} \end{cases} \tag{4}$$

$$D \cdot y = 0 \tag{5}$$

$$D \cdot z = 0 \tag{6}$$

$$A(x,y,z) = \begin{cases} 1 & x < 0.5-\sigma \vee x > 0.5+\sigma \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Note how the displacement is C^0 continuous, important for application in the GPU. That means that the displacement is never undefined for any position, including those in the cut region. The alpha mask, which is zero in the section that is cut, takes care of the discontinuity. Modifying the parameters of the knife, to simulate different angles and gap sizes, can be done by scaling and rotating the mask, i.e., by applying an affine transformation to D and A simultaneously. This displacement can be pre-computed on a 3D texture, or sampled on-the-fly on a shader. We opted for the first approach, using 3D framebuffer objects to create a displacement directly on the GPU, and use the 3D texture as a look-up table when rendering the deformed volume. Computing the displacement on the fly scales with the number of samples created for a given volume, which may have an impact in performance as the deformation becomes more complex. In the spirit of generality, our approach works without much knowledge of the type of deformation, except that it is sampled as a 3D displacement field.

3. Algebraic operations on displacement maps

One of the advantages of displacement maps is the ability to operate on them algebraically. This enables us to create complex deformations from the combination of simple primitive displacements, and is the basis for constrained illustrative deformation. This section describes several mechanisms for operating displacement maps, such as modulation, composition and addition.

3.1. Modulation

This is a unary operation which scales the effects of a displacement based on a scalar field, which is called the *modulation field*. Unlike affine transformations, the modulation is in general not constant. We can define this in two ways, depending on the space in which modulation is defined. The first one is modulating in *displacement* space, which is the most common interpretation of modulation. In this case, the displacement is defined as

$$\mathbf{p}' = \mathbf{p} + C(\mathbf{p})D(\mathbf{p}) \tag{8}$$

where C is a scalar field.

Normals can be obtained by evaluating the Jacobian of the constrained displacement. According to vector calculus [36], for a vector field G, defined as the product of vector field F and a scalar field f, $G = fF$, the jacobian J_G of G is derived from the product of derivatives as follows:

$$\mathsf{J}_G = f\,\mathsf{J}_F + F\nabla_f^{\top} \tag{9}$$

where J_F is the Jacobian of the vector field F and ∇_f is the gradient of scalar field f. Therefore, the normal estimation for modulated deformation can be defined as

$$\vec{\mathbf{n}}^{(p)} = (\mathsf{I} + C\mathsf{J}_\mathsf{D} + D\nabla_C^{\top})_{(p)}^{\top}\,\vec{\mathbf{n}}^{(p')} \tag{10}$$

One of the limitations with this approach is the assumption of linearity of the displacements implied by the modulation process. This may cause unrealistic deformations for non-linear displacements, such as twisting, where decreasing the magnitude of a large twist angle does not represent a less pronounced twisting. For this reason, we propose a different modulation mechanism, which modulates in *deformed* space instead of displacement space. It acts as a pre-warping of the space.

This modulated displacement is then defined as

$$\mathbf{p} = \mathbf{p}' + D(T(\mathbf{p})) \tag{11}$$

where $T : R^3 \mapsto R^3$ is a space transformation. Unlike the first interpretation, here the scalar field is used to warp the points in the deformed space, instead of the resulting displacement. This approach, however, only applies to certain displacements, which we call *self-scaled*, that have the property that the displacements at different scales appear in the displacement field at different locations. An example is a twisting displacement. Incrementing the magnitude of the twisting is equivalent to translate the twist displacement. In this case, if the twist occurs along the Z direction, the transformation can be defined as

$$T(\mathbf{p}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & C(\mathbf{p}) \end{pmatrix} \mathbf{p}$$

Although this is limiting, it enables the generation of complex deformations with little cost. Normal estimation for instance is simpler, as it can be done by concatenating the transpose of the Jacobians

$$\mathsf{J}_B = \mathsf{I} + \mathsf{J}_D(T(\mathbf{p}))\mathsf{J}_T(\mathbf{p}) \tag{12}$$

One of the advantages of this approach is that it works for cuts as well, as there is no need for specifying additional modulation for the alpha mask. It also enables the introduction of more complex constraints such as rigid movement of features. Fig. 4 shows two twisting operators applied to the bar dataset. For the first row, the modulation is defined as the distance along the z direction, which is equivalent to define an *anchor* or constraint at one end of the bar. For the second row, we define a modulation function such that an entire region is moved rigidly. The corresponding modulation functions are shown on the right side of Fig. 4.

3.2. Addition

Displacement maps offer the flexibility of creating new displacements via addition. Let D_1 and D_2 be two displacement maps. Then, a new displacement map $D_S = D_1 + D_2$ can be defined, such that

$$\mathbf{p} = \mathbf{p}' + D_1(\mathbf{p}') + D_2(\mathbf{p}') \tag{13}$$

Further, since the displacements contain discontinuities, the new alpha mask is defined as

$$A_{1+2} = \rho \min(|A_1|, |A_2|) \tag{14}$$

$$\rho = \begin{cases} -1 & A_1 < 0 \text{ or } A_2 < 0 \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

Addition can be performed on a pre-processing stage to create a new displacement and then apply it to a volumetric object. Alternatively, they can be added on the fly. In that case, the normal transformation can be obtained by simply adding the Jacobians of the displacements

$$\overrightarrow{\mathbf{n}}^{(p')} = (\mathrm{I} + \mathrm{J}_{D1}^{(p')} + \mathrm{J}_{D2}^{(p')})^{\top} \overrightarrow{\mathbf{n}}^{(p)} \tag{16}$$

where $\mathrm{J}_{D\ 1}$ is the Jacobian of D_1 and $\mathrm{J}_{D\ 2}$ is the Jacobian of D_2.

3.3. Composition

In general, two transformations can be combined as follows:

$$\mathbf{p} = G_1(G_2(\mathbf{p}')) \tag{17}$$

where $G_1(\mathbf{u}) = \mathbf{u} + D_1(\mathbf{u})$ and $G_2(\mathbf{v}) = \mathbf{v} + D_2(\mathbf{v})$ are displacement mappings.

For computing the normal, we must simply concatenate the Jacobians of the two mappings

$$\overrightarrow{\mathbf{n}}^{(p')} = (\mathrm{J}_{G1} \times \mathrm{J}_{G2}) \overrightarrow{\mathbf{n}}^{(p)} \tag{18}$$

Since each displacement changes the frame of reference, composition is, unlike addition, not commutative in general. An example is shown in Fig. 5 where a bending and a twisting displacement are applied to a bar dataset in different order.

4. Constrained illustrative deformation

One of the challenges of volume deformation has been the inclusion of semantic information. For example, to obtain visually acceptable illustrations, deforming tissue should not intersect other non-deforming tissue. In other cases, it becomes necessary to separate different materials, so that certain objects deform rigidly while the surrounding tissue deforms elastically. Here, we show that these constraints can be encoded as operations on displacement fields with little computational cost. This is a fast alternative to physically based simulation. Constrained deformation can be obtained by modulating and adding displacements, as described in Section 3. Modulation plays an important role for defining geometric and spatial constraints on the displacement, as described below.

4.1. Collision-free volume manipulation

With our approach, we can introduce collision constraints via modulated displacement. In this case, we can defined the scalar field $C : \mathbb{R}^3 \to \mathbb{R}$ in Eq. (8) as a function of the one-sided distance transform to certain feature of interest.

$$C(\mathbf{p}') = \frac{1}{\max(D(\mathbf{p}'))} DT(\mathbf{p}') \tag{19}$$

where $DT(\mathbf{p})$ is a scalar field representing the distance field to a feature of interest. This definition has two properties:

(1) Elements within the feature of interest are not deformed. This occurs since the one-sided distance field has value $C(\mathbf{p}) = 0$ inside the feature, and therefore the displacement is the zero vector.
(2) Elements outside the feature of interest cannot be deformed through the region of interest. This happens because a point $\mathbf{p}'$

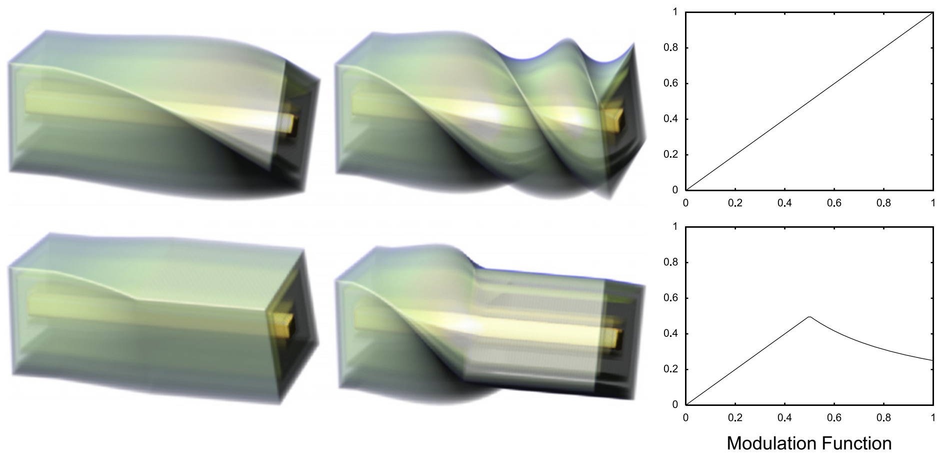
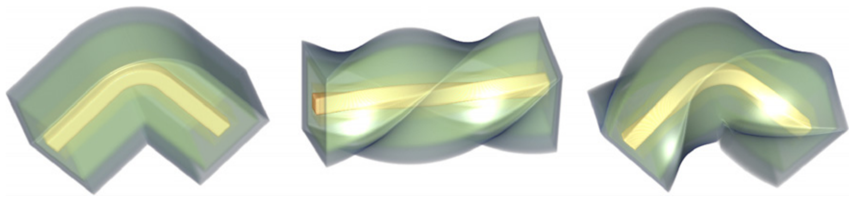


Fig. 4. Modulated deformation of bar dataset.



$\mathbf{p}' = bend(\mathbf{p})$ $\mathbf{p}' = twist(\mathbf{p})$ $\mathbf{p}' = bend(twist(\mathbf{p}))$

Fig. 5. Composition of two displacement maps, bending and twisting.

in the sampled space can only be moved within a sphere of radius r, defined as

$$r = \mathbf{p} - \mathbf{p}' = C(\mathbf{p}')D(\mathbf{p}') = \frac{1}{\max(D(\mathbf{p}'))} DT(\mathbf{p}')D(\mathbf{p}') \le DT(\mathbf{p}')$$

We used this method to represent collision-free deformation on the knee (Fig. 1). In this case, we apply a deformation to the back part of the knee, while keeping the bone tissue untransformed. With homogeneous volume deformation, the bone tissue is transformed with the rest of the volume. A method for constraining the effect of deformation on certain features was proposed before as feature-aligned deformation [1]. Although it preserves the bone tissue, the deformed tissue passes through the bone in an unrealistic manner. With our constrained approach, the deformation is guaranteed to prevent self-collision.

Another example is shown in Fig. 6, where we constrain the deformation to a small window illustrating a cut on the skin. It is natural that deformed tissue should not intersect the undeformed tissue such as skin. Fig. 6(a–c) shows three stages of constrained deformation of one of the muscles and veins, after being cut. We can see that the deformed tissue is constrained to the volume of the cut. In Fig. 6(d), we see that unconstrained deformation appears unrealistic, as veins are deformed through the skin. Fig. 6(e) shows the effect of changing the constraint volume C. Note how the deformation adapts to the new shape. Fig. 6(f) shows yet another cut and the constrained illustrative deformation of the veins.

4.2. Heterogeneous deformation

The ability to combine and modulate displacements allows us to deform heterogeneous materials. This has been a major challenge in volume deformation. With our approach, we can accomplish this with modulated displacement fields over different regions in a volume.

For example, we can simulate complex materials that combine rigid and non-rigid materials. Let us consider the simple case of one object moving rigidly embedded in an elastic medium. Similar to the one above, let us define a scalar field C as a one-sided distance function to the rigid material. If we consider *forward* transformation, the final position of a point $\mathbf{p}'$ is

$$\mathbf{p} = \mathbf{p}' + C(\mathbf{p}')D_F(\mathbf{p}') + (1 - C(\mathbf{p}'))(R(\mathbf{p}') - \mathbf{p}')$$

where D_F is a forward displacement and R is a rigid transformation (i.e., a rotation and a translation). We can see that, inside the rigid object, i.e., when $C(\mathbf{p}') = 0$, the deformation is $\mathbf{p} = R(\mathbf{p}')$. For volume deformation, however, we must invert that expression, whose close form may be difficult to obtain. Instead, we may assume that the displacement and the rigid transformations are infinitesimally small. In that case, $D_F(\mathbf{p}') \approx D(\mathbf{p})$, where $D(\mathbf{p})$ is the inverse displacement of D_F, and $\mathbf{p}' \approx R^{-1}(\mathbf{p})$, where R^{-1} is the inverse rigid transformation. The inverse transformation is then

$$\mathbf{p}' = \mathbf{p} + C(R^{-1}(\mathbf{p}))D(\mathbf{p}) + (1 - C(R^{-1}(\mathbf{p})))(R^{-1}(\mathbf{p}) - \mathbf{p})$$

Fig. 7 shows the bending of a bar, where a portion of the inner core is considered as rigid. The constraint field C is defined as the one-sided distance field to the inner core. D is an inverse bending field, which encodes a sufficiently small bending transformation (typically of an angle of about one degree). The rigid transformation R is a rotation about the bending center. Note how the core bends rigidly, while the outer layer follows the original displacement. Different bending deformations are obtained using path integration.

Fig. 8 shows an illustration of a whiplash deformation, here simulated as a bending operation. To simulate the different stages of a bending, we use path integration from the initial bending (left) towards the final bending (right) using a displacement field that encodes a small bending displacement and the addition operation. Constrained deformation is used to move the skull rigidly. In this case, we define the constrain volume C as a distance field to the skull mask, which we use for interpolating between the bend and rigid displacement.

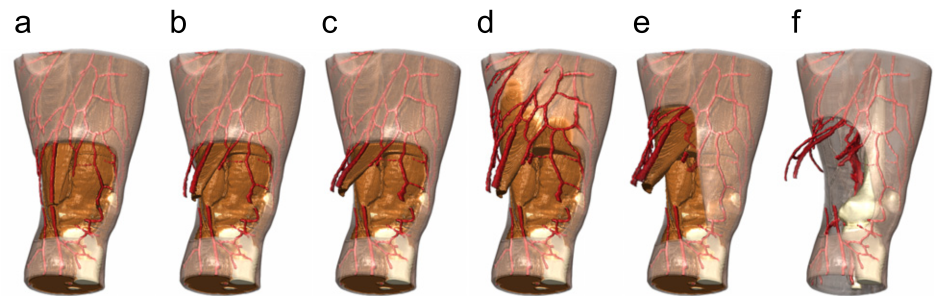


Fig. 6. Constrained deformation of one leg of the segmented visible human data set. (a–c) Sequence of deformation of muscle and veins, constrained to the cut region in the skin; (d) unconstrained deformation results in portions of the muscle and vein to pass through the skin unrealistically; (e) different constraint volume. Note that the deformation adapts accordingly to avoid intersection of the tissues; (f) constrained deformation on the veins.

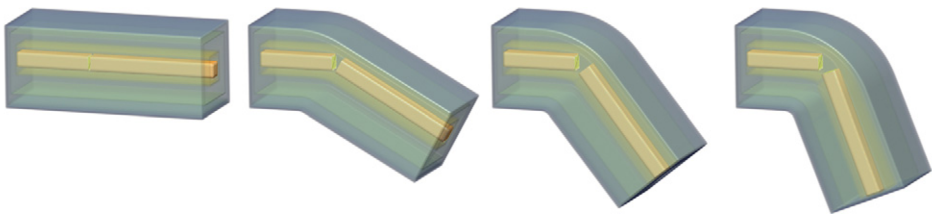


Fig. 7. Deformation of heterogeneous volumes. In this data set, the bar is deformed with a bending operation while constraining the inner core (yellow) to more rigidly. This is accomplished with displacement modulation, where the distance to the rigid rod defines an interpolation factor that smoothly blends from the rigid displacement to the bending displacement. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

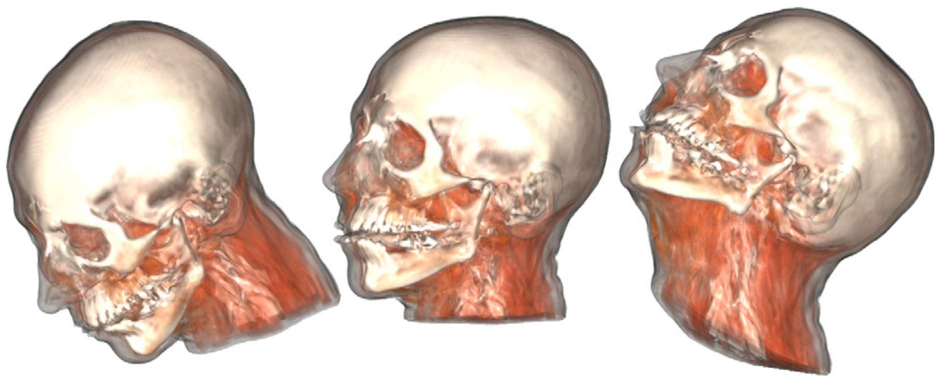


Fig. 8. Illustrative deformation of the CT head dataset to depict a whiplash injury. Inspired by medical illustrations, this deformation is obtained by applying a bending operation to the CT head dataset. We use constrained deformation to simulate the rigidity of the skull, without the need for complex finite element interactions.

Table 1
Cost of constrained deformation, in rendering time (seconds per frame).

	Undeformed	Unconstrained	Constrained (proc.)	Constrained (sample)
No lighting	0.024	0.037	0.040	0.053
Lighting	0.100	0.207	0.210	0.310

We compare undeformed volume rendering vs. deformation, which can be unconstrained, constrained using a procedural function C, or constrained using a sampled volume.

4.3. Limitations

One of the advantages of our approach is that deformations can be unified in a single representation, which are displacement fields. However, in practice, they must be stored in a discrete and finite data structure. In our case, we store these fields as 3D textures. Accessing the displacement field from a texture, as well as the constrains, imposes additional costs in rendering. Table 1 summarizes the cost of rendering a volume of dimensions $256 \times 256 \times 256$ under no deformation, unconstrained and constrained deformation. We used an nVidia GTX 285 card with 2GB of texture memory and rendering on a 512×512 viewport. If no lighting is required, the overhead is negligible. However, when lighting is required, the on-the-fly computation of gradients and Jacobian matrices become costly. One way to alleviate this problem is to pre-compute the Jacobian and gradients in a previous pass (using framebuffer objects to keep the computation in the GPU) and then use the pre-computed information to compute the lighting.

The sampling of displacements in a 3D texture also poses a problem for those points outside the boundary of the 3D textures. Since no information is available outside the displacement field, they are not deformed, or they are deformed in an undesired way. For this reason, it is expected that displacements should be sampled for all points in the boundary box of the *deformed* space. This, however, may lead to large displacement fields. Several mechanisms to deal with this problem can be devised. One is the use of scale factors to increase the spatial extents of the displacement domain. Another is with a partitioning of the domain, where each partition is assigned a different displacement field. This partitioning follows closely the convex hull of the deformed object and therefore makes better use of texture memory for large datasets. The exploration of a general solution is ongoing research.

Constrained deformation is useful for preventing intersection of a volume with pre-determined regions, such as anatomical features. However, it cannot be used to prevent self intersection of the deformed volume with other deformed regions. Since displacements are sampled in a continuous volume, self-intersection of a deformed volume cannot occur in the first place, but it will create unrealistic bends and duplication of voxels for complex deformations. Since our approach is based on templates and not direct manipulation, once a template is ensured to be non-intersecting, further operations will generate non-intersecting displacements. A general solution to this problem is ongoing research.

5. Conclusions

We have presented a method for simulating physical constraints in volume deformation without expensive simulation or complex meshing. Instead, we use algebraic operations on displacement maps to mimic the effects of deformations when considering certain physical properties, such as collision-free deformation, rigidity and elasticity. This paradigm is a departure from physically based simulation, usually a bottom-up approach that models forces and velocities on individual points to derive complex deformations. Instead, we advocate a top–down approach, where we consider high-level transformations, such as bending, twisting and cutting, in the presence of high-level constraints, such as rigidity or collision, for the purposes of illustration. We believe this approach is a faster and more effective mean to illustrative deformation, that overcomes the issues and costs involved in preparing medical data sets, such as segmentation, meshing and simulation. Our approach has applications beyond medical illustration and animation, such as interactive data exploration and visualization.

References

[1] Correa C, Silver D, Chen M. Feature aligned volume manipulation for illustration and visualization. IEEE Transactions on Visualization and Computer Graphics (Proceedings of visualization/ information visualization 2006) 2006;12(5):1069–76.

[2] McGuffin MJ, Tancau L, Balakrishnan R. Using deformations for browsing volumetric data. In: Proceedings of IEEE visualization (VIS) 2003, 2003. p. 401–8.

[3] Correa CD, Silver D, Chen M. Discontinuous displacement mapping for volume graphics. In: Proceedings of the fifth eurographics/IEEE VGTC workshop on volume graphics 2006, Eurographics Association, 2006. p. 9–16.

[4] Chen W, Lu A, Ebert DS. Shape-aware volume illustration. Computer Graphics Forum 2007;26(3):705–14.

[5] Rautek EGP, Bruckner S. Illustrative visualization: new technology or useless tautology. ACM SIGGRAPH Computer Graphics 14(8).

[6] Ebert D, Rheingans P. Volume illustration: non-photorealistic rendering of volume model. In: IEEE visualization, 2000. p. 195–202.

[7] Lu A, Morris CJ, Taylor J. Illustrative interactive stipple rendering. IEEE Transactions on Visualization and Computer Graphics 2003;9(2):127–38.

[8] Viola I, Kanitsar A, Groller ME. Importance-driven feature enhancement in volume visualization. IEEE Transactions on Visualization and Computer Graphics 2005;11(4):408–18.

[9] Wang L, Zhao Y, Mueller K, Kaufman AE. The magic volume lens: an interactive focus+context technique for volume rendering. In: IEEE visualization, 2005. p. 47.

[10] Kruger J, Schneider J, Westermann R. Clearview: an interactive context preserving hotspot visualization technique. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):941–8.

[11] Bruckner S, Gröller ME. VolumeShop: an interactive system for direct volume illustration. In: Silva HRCT, Gröller E, editors. Proceedings of IEEE visualization 2005, 2005. p. 671–8.

[12] Bruckner S, Groller ME. Exploded views for volume data. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):1077–84.

[13] Correa C, Silver D, Chen M. Illustrative deformation for data exploration. IEEE Transactions on Visualization and Computer Graphics (Proceedings of visualization/information visualization 2007) 2007;13(6):1320–7.

[14] Li W, Agrawala M, David Salesin BC. Automated generation of interactive 3D exploded view diagrams.

[15] Chen M, Correa C, Islam S, Jones W, Shen P, Silver D, Walton SJ, Willis J. Manipulating, deforming and animating sampled object representations. Computer Graphics Forum 2007;26(4):824–52.

[16] Nealen A, Muller M, Keiser R, Boxerman E, Carlson M. Physically based deformable models in computer graphics. In: Eurographics STAR report, 2005.

[17] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. In: SIGGRAPH '86: Proceedings of the 13th annual conference on computer graphics and interactive techniques, 1986. p. 151–60.

[18] Barr AH. Global and local deformations of solid primitives. In: SIGGRAPH '84: Proceedings of the 11th annual conference on computer graphics and interactive techniques, 1984. p. 21–30.

[19] Kho Y, Garland M. Sketching mesh deformations. In: SI3D '05: Proceedings of the 2005 symposium on interactive 3D graphics and games, 2005. p. 147–54.

[20] Lewis JP, Cordner M, Fong N. 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, 2000. p. 165–72.

[21] Correa CD, Silver D. Programmable shaders for deformation rendering. In: GH '07: Proceedings of the 2007 ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware, 2007. p. 89–96.

[22] Hughes JF. Scheduled Fourier volume morphing. In: SIGGRAPH '92: Proceedings of the 19th annual conference on computer graphics and interactive techniques, 1992. p. 43–6.

[23] He T, Wang S, Kaufman A. Wavelet-based volume morphing. In: VIS '94: Proceedings of the conference on visualization '94. Los Alamitos, CA, USA: IEEE Computer Society Press; 1994. p. 85–92.

[24] Lerios A, Garfinkle CD, Levoy M. Feature—based volume metamorphosis. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on computer graphics and interactive techniques, 1995. p. 449–56.

[25] Fang S, Raghavan R, Richtsmeier JT. Volume morphing methods for landmark-based 3D image deformation. In: Loew MH, Hanson KM, editors. Proceedings of SPIE, medical imaging 1996: image processing, vol. 2710, 1996. p. 404–15.

[26] Gagvani N, Silver D. Animating volumetric models. Graphical Models 2001;63(6):443–58.

[27] Kurzion Y, Yagel R. Interactive space deformation with hardware-assisted rendering. IEEE Computer Graphics and Applications 1997;17(5):66–77.

[28] Chen H, Hesser J, Manner R. Ray casting free-form deformed-volume objects. The Journal of Visualization and Computer Animation 2003;14:61–72.

[29] Islam S, Silver D, Chen M. Volume splitting and its applications. IEEE Transactions on Visualization and Computer Graphics 2007;13(2):193–203.

[30] Chen M, Silver D, Winter AS, Singh V, Cornea N. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In: Proceedings of volume graphics '03, 2003. p. 35–44.

[31] Gibson SF. 3D chainmail: a fast algorithm for deforming volumetric objects. In: SI3D '97: Proceedings of the 1997 symposium on interactive 3D graphics, 1997. p. 149.

[32] Westermann R, Rezk-Salama C. Real-time volume deformations. Computer Graphics Forum 20(3).

[33] Rezk-Salama C, Scheuering M, Soza G, Greiner G. Fast volumetric deformation on general purpose hardware. In: HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on graphics hardware, 2001. p. 17–24.

[34] Singh V, Silver D, Cornea N. Real-time volume manipulation. In: Proceedings of volume graphics '03, 2003. p. 45–51.

[35] Walton S, Jones M. Volume wires: a framework for empirical non-linear deformation of volumetric datasets. Journal of WSCG 2006; 14.

[36] Davis H. Introduction to vector analysis, 3rd ed. Allyn and Bacon; 1967.