Feature Aligned Volume Manipulation for Illustration and Visualization

Carlos D. Correa, *Student Member, IEEE*, Deborah Silver, *Member, IEEE*, and Min Chen

Abstract—In this paper we describe a GPU-based technique for creating illustrative visualization through interactive manipulation of volumetric models. It is partly inspired by medical illustrations, where it is common to depict cuts and deformation in order to provide a better understanding of anatomical and biological structures or surgical processes, and partly motivated by the need for a real-time solution that supports the specification and visualization of such illustrative manipulation. We propose two new feature-aligned techniques, namely surface alignment and segment alignment, and compare them with the axis-aligned techniques which was reported in previous work on volume manipulation. We also present a mechanism for defining features using texture volumes, and methods for computing correct normals for the deformed volume in respect to different alignments. We describe a GPU-based implementation to achieve real-time performance of the techniques and a collection of manipulation operators including peelers, retractors, pliers and dilators which are adaptations of the metaphors and tools used in surgical procedures and medical illustrations. Our approach is directly applicable in medical and biological illustration, and we demonstrate how it works as an interactive tool for focus+context visualization, as well as a generic technique for volume graphics.

Index Terms—Illustrative visualization, Illustrative manipulation, GPU computing, volume rendering, volume deformation, computer-assisted medical illustration

1 Introduction

In science, medicine and engineering, hand-drawn illustrations often include manipulating part of an object to depict the stages and outcome of a procedure, uncover hidden features, or reveal the spatial relationship between different components of the objects. Such manipulation typically includes the following characteristics:

- It often contains *cuts and dissections*, which, for example, are commonly found in illustrations of surgical procedures as exemplified by Figure 1(b).
- It may allow *feature-sensitive operations*, which can be applied to a semantic component of the object, such as the skin in Figure 1(b), without affecting other parts of the object.
- It may enable *ubiquitous operations*, which can be applied to various parts of the object with different geometric transformations, as shown in Figure 1(d).
- It can facilitate *virtual operations*, which do not necessarily conform to the reality, such as the unreal flaps used to illustrate anatomical structure in Figure 1(d).

In the context of *illustrative visualization*, we refer to such manipulation as *illustrative manipulation*, which provides a means for specifying and realizing visualization that contains cuts, dissections, distortion, and various other forms of deformation that are not present in the original data. For example, in Figure 1(a), the skin layer (as the context) of the hand is manipulated using the metaphor of "retractors" to reveal the bones or vessels (as the focus), facilitating an illustrative visualization with focus+context. In Figure 1(c), a number of virtual manipulation operations are applied to the visible human dataset to achieve an illustrative visualization in a manner similar to a classical anatomic illustration, without resorting to the complex and sometimes contentious processes employed by some contemporary exhibitions.

It is, however, not intended to provide a physically-based simulation of the internal and external forces involved in the deformation or the physical interaction between the tools and different parts of the object. Although physically-based modeling is critical to applications such as surgical simulation, and could ultimately be desirable for illustrative manipulation, a huge computational cost is necessary for any realistic modeling and simulation of a combination of physical behaviors such as elastic and plastic deformation, fractures, stress-strain curves of complex materials, including skins, soft tissues, body fluids, etc. Hence, for illustrative manipulation, one must give the priority to the *interactivity* and *operatability* in the process of realizing illustrative visualization.

In this paper, we propose a feature-based technique for manipulating volumetric objects for illustrative visualization, and describe a GPU-based implementation that enables interactive specification and rendering such visualization. Inspired by medical illustrations that frequently depict the results of manipulation with tools such as peelers, retractors and pliers, we allow the specification of manipulation through a collection of procedurally-defined *manipulation operators*. Because we place a significant emphasis on the interactivity and operatability, we need to address a number of conflict technical requirements, for example, (i) between the axis-aligned volume texture storage and the arbitrary geometry of features, and (ii) between the normals on a cut surface and the gradient-based normals within the original volume.

When specifying a cut or peel, one important consideration is "alignment". Here, *aligned manipulation* refers to those cuts or peels that are applied to certain layers while other features of interest are preserved. The simplest is *axis-aligned*, which aligns the operator with the axis-planes. It is feature-insensitive and is applied to all points within the volume bounds. This however is not always satisfactory as the "object" within the volume is not necessarily cubic. We introduce two feature-based methods, namely *surface alignment* and *segment alignment* for modeling and rendering illustrative manipulation, and compare them with the traditional *axis alignment* method. We devise a method for estimating accurate normals along the surface of cuts and dissections while maintaining continuous normals, which allows us to obtain correct shading of the object being manipulated, opaque or translucent.

We built our approach on the general concept of space warping [1], which was traditionally realized using ray casting without interactivity. We provide an efficient GPU implementation for allowing illustrative manipulation to be specified and rendered at interactive rates, and we demonstrate the technical feasibility and usability of our approach by applying various illustrative manipulation to a number of volume datasets interactively on a Pentium XEON 2.8 GHz PC with a Quadro

- *Carlos D. Correa and Deborah Silver are with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, E-mail: cdcorrea, silver@caip.rutgers.edu*
- *Min Chen is with Department of Computer Science, University of Wales, Swansea, E-mail: m.chen@swansea.ac.uk*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

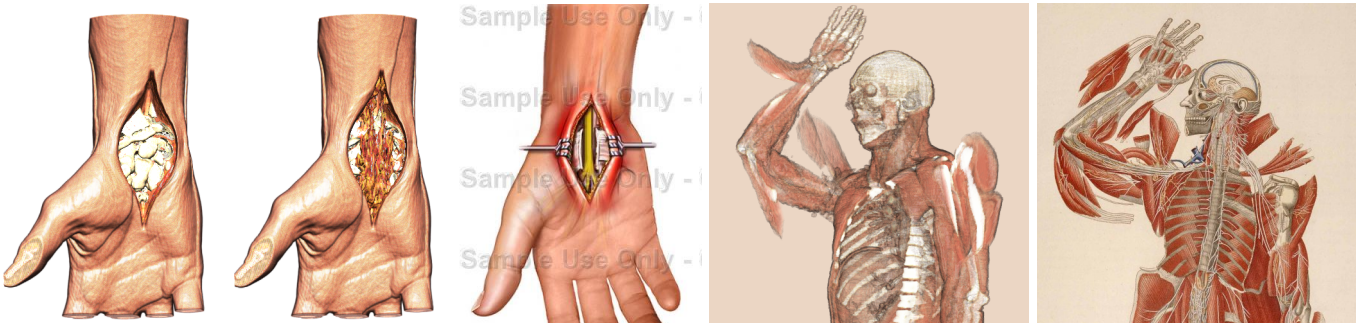


Fig. 1: Feature-aligned volume manipulation (a) A feature-aligned retraction applied to a human hand data set, showing bones (left) and vessels (right) (b) Surgical illustration of a hand (Copyright ©2006 Nucleus Medical Art. All rights reserved. www.nucleusinc.com) (c) Multiple peel and cutting away operators applied to the visible human data set (d) Illustration of human anatomy with dissected "flaps", by Antonio Scrantoni and Paolo Mascagni, 1833 (U.S. National Library of Medicine), similar to exhibitions such as BodyWorlds [22], and Bodies, the Exhibition [9].

FX4400 graphics card.

2 Related Work

Illustrative visualizations are often associated with cutaway visual effects [7] and non photorealistic rendering [8]. This paper is concerned with the former aspect of illustrative visualization, and in particular, we are interested in creating feature-based cutaway visual effects through interactive manipulation of volume datasets.

McGuffin *et al.* [15] proposed a set of interactive manipulation widgets to browse volume data using *forward mapping*. They used individual voxels as rendering primitives, which can be directly transformed, allowing the modeling of effects such as cuts and openings. While voxels were grouped into segments as features for manipulation, the rendering did not involve connectivity information. It was thereby difficult to achieve good quality visualization, and aliasing was noticeable especially in close-up views or with large deformation such as stretch.

An alternative approach is to implement manipulation operations using *backward mapping*. The basic idea, space warping, was first outlined by Barr [1], and was realized for volume manipulation through the notion of ray deflectors [13] and spatial transfer functions [5, 12]. Both [13] and [5] employed ray casting for sampling the deformed space, hence high quality rendering was attainable. However, both provided only a brute-force solution to compute the normals of the deformed or dissected objects, and neither considered feature-sensitive manipulation. Furthermore, due to the computational costs involved, this ray-casting approach for volume manipulation is not yet able to support interactive manipulation on current desktop computers.

There has been a series of efforts for achieving interactive volume rendering by exploiting GPU capacities (e.g., [24, 18, 21]). Westermann and Salama [25] utilized 3D texture mapping hardware to achieve interactive deformation of volume datasets. Rezk-Salama *et al.* [19] used general purpose rendering hardware, coupled with edge and face constraints, to facilitate the adaptive subdivision of piecewise patches of a volume object. Singh and Silver [20] used mid-plane geometry to specify and render deformation about joints of visible human, in conjunction with texture mapping hardware. In order to obtain normals that correspond to the deformed volume, Westermann and Salama [25] employed an approximate solution based on a two-pass rendering approach [17]. Rezk-Salama et al. [19] obtain normals via a linear approximation of the gradient. Chen et al. [3] use raycasting and employ inverse mapping to achieve deformation. Weiskopf *et al.* [23] presented a technique for estimating the normals of clipped volumes with a weighting function for blending the normals near the boundaries of the clipping plane. Correa et al. [6] employed generalized displacement mapping as means to achieve real-time manipulation of volume data. Nevertheless, [25], [19] and [3] implemented mainly continuous deformation without cuts, while [23] and [6] considered mainly axis-aligned operators.

The work presented in this paper brings together the desirable elements of the above-mentioned previous work. We draw the feature-based approach from [15] in order to create more meaningful illustrative visualization. We draw the backward mapping approach from [13, 5] in order to facilitate complex manipulation operators and maximize the rendering quality at interactive speeds. We extended the use of segment based features in [15] by introducing manipulation based on surface features. We further extended the normal calculation method in [23] by allowing normal adjustment along features of interest. For the first time, we have delivered complex visual effects, such as feature-based peeling and opening, interactively with a rendering quality comparable to non-interactive solutions such as [5]. As mentioned in Section , this work is not intended to provide a physically-based simulation of various manipulation, for which there is a rich collection of literature ([11, 4, 16]).

3 Manipulation Operators

A *manipulation operator* is a metaphor for a transformation mapping from the original volumetric space to a dissected or deformed volumetric space. From the perspective of a user, such an operator functions like a surgical tool, such as a pair of retractors, and can be used to perform an illustrative manipulation on a volumetric model. In this work, we designed and implemented four types of operators, namely

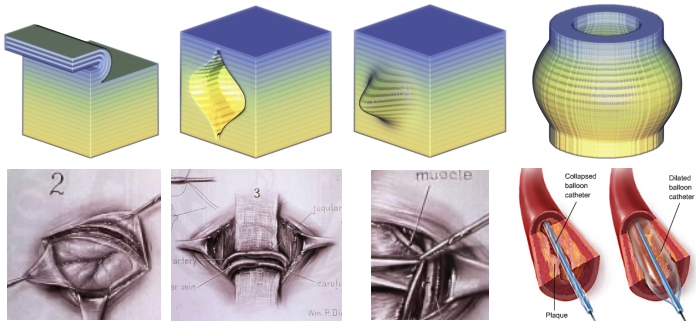


Fig. 2: Iconic representations of four manipulation operators, and example medical illustrations showing similar metaphors. Drawings courtesy of the U.S. National Library of Medicine. Dilator illustration is copyright ©medmovie.com.

peeler, retractors, pliers and dilator. Figure 2 shows the iconic representation of these operators as applied to a simple volume object, and each is accompanied by a medical illustration that exemplifies the operator. All four operators are procedurally defined, and each is associated with a set of parameters, such as operatable size, distortion scale, etc., which can be changed interactively. These operators can be placed anywhere on or within the volumetric model. The transformation of the operators over the volume model or its features is determined dynamically during rendering (see 3.2). The detection of self-collision between different parts of the volume model is not featured in the current implementation, because of the computational cost for providing a meaningful feedback rather than the cost for detection. Nevertheless, self-collision can be prevented by interactively modifying the parameters of a manipulation operator. Below we describe briefly the functionality of the four operators, and further examples of their use can be found in Section 6.

Peeler. The peeler simulates peeling or cutting of the outer layers of a volumetric model. In Figure 3, a peeler with cylindrical fold-back is shown applied to the CT head dataset. Different parameters of the peeler, such as thickness of the cut and the angle of bending of the peeled layer, can be manipulated interactively to obtain different illustrative effects. For example, the different virtual flaps in Figure 1(c) are obtained by placing multiple peeler operators around the body and varying their parameters.

Retractors. In the context of surgical illustration, retractors are tools used to spread organs or bones, or to hold back soft tissues such as skin. In the context of illustration, they are useful for illustrating the access to the internal organs.

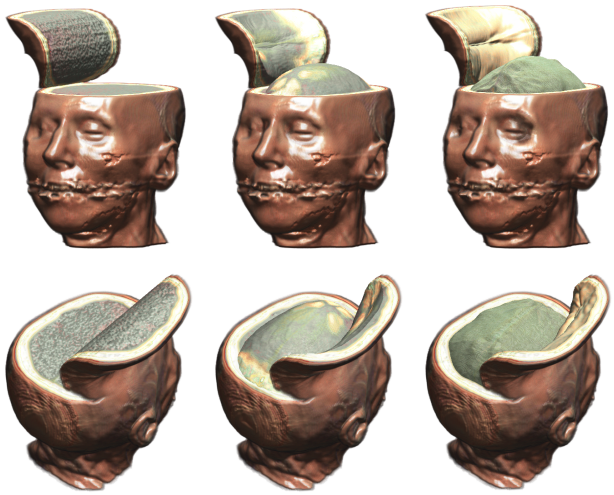
Pliers. Pliers are tools used to grasp tissue and pull it or poke it into the volumetric object. The operator parameters include the shape of the displacement and the radius of influence which specifies how the displacement propagates through the volume. In Figure 2, the pliers are shown pulling a blood vessel.

Dilator. Dilators are used to gain access into narrow regions, cuts or vessels. They essentially scale the region from the inside typically by blowing air, to increase the visibility or accessibility of the region. When applied to volumetric manipulation, dilators have a similar effect to that of volumetric magic lenses [14].

3.1 Manipulation Alignment

When using the manipulation operators, we define two properties: *alignment* and *placement*. Alignment here refers to the preservation of interesting features while operating, giving the impression that it is aligned with features. Placement, on the other hand, refers to the movement in 3D space of the operator. For example, in Figure 3(c), the operator is said to be aligned with the brain tissue, since the peeler appears to follow its contour. Placement of the operator defines how deep is the cut, or in which direction. This paper addresses the problem of *alignment*. Manipulation operators can be aligned with one of the three main axes, which are intuitive to place in three dimensional space, and easy to implement in conjunction with a texture-based volume renderer. Axis alignment is usually limited, since very rarely features of interest are planar. This limitation can be seen in Figure 3(a). In this paper, we propose two *feature aligned* techniques for allowing manipulation operators that are sensitive to the specific features on a volume model. We consider surface features that are defined by an iso-surface, and volume features that are define by a segment, resulting in the notions of *surface-aligned* and *segment-aligned* manipulation operators respectively. The merits of such alignments can be seen Figures 3(b) and 3(c).

Consider the skin of the head as the *context*, and the brain as the *focus* of these visualizations. From Figure 3(a) where a peel is performed using axis-alignment, we can observe that the part being peeled is wedge shaped and the operator cuts through the brain. The visualization conveys limited information about the part in focus. Feature alignment



(a) Axis Alignment (b) Surface Alignment (c) Segment Alignment

Fig. 3: An example of different types of alignment. (a) Axis aligned peel. Note how the peeled layer is thick and flat, since it is aligned with an orthogonal axis. (b) Surface aligned peel, aligned with a computed distance field. Notice how it approximates a surface. (c) Segment aligned peel, based on segmentation, which is more accurate. Note that in the feature based alignment (b) and (c) the peel is thin and rounded.

corrects this problem. Virtual surfaces are created without segmentation (except for background), using a distance field computation. The distance field is computed based on a boundary, from which virtual shells of different thicknesses can be defined. For example, in 3(b), a surface-aligned peel is applied to the top of the head with a uniform depth from the surface of the skin. Such alignment can be used to investigate and illustrate layered structures without the pre-knowledge of segmentation. However, while this is a good approximation , the operator still cuts through the brain. If we have the specification of volume features, typically obtained from segmentation or defined by a range of iso-values, we can create a more effective focus+context visualization. In 3(c), with the segmentation knowledge of external layers including the skin and skull, we apply a segment-aligned peel to the head. The brain, that is, the focus, is not only highlighted but also visualized with correct geometry. In addition, the correct shading at the back of the peel provides further meaningful context to the visualization.

3.2 Volume Manipulation

In this work, *volume manipulation* is defined as point-wise mapping that transforms positions of all points in an axis-aligned bounding volume V to new positions. Let P_V be a set of all points in V. For each point $p \in P_V$.

$$p' = T_F(p) \qquad p = T_F^{-1}(p') \tag{1}$$

where T_F denotes a *forward transformation* and T_F^{-1} the corresponding *inverse transformation*. Hence, we obtain a new set $P'_V = \{p' | \forall p \in P_V, p' = T_F(p)\}$. Let V' be the new axis-aligned bounding volume for all the points in P'_V. Unlike the majority of previous work on volume deformation, T_F and T_F^{-1} are not necessarily continuous. While the forward transformation is intuitive, it has several shortcomings. First, it is difficult to know the boundary of the new volume V' after the manipulation without evaluating T_F analytically. One solution to this difficulty is for the user to specify such a bounding volume. Second, it is difficult to render a manipulation operator that involves cuts and dissections. Consider a retractor and a dilator. The former spreads the points apart and introduces empty space between the transformed points, while the latter stretches the points but maintains continuity

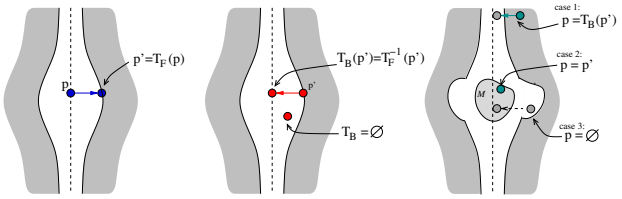


Fig. 4: Forward (left) and backward (middle) transformation of a retractor tool, modeled as a displacement in the direction of the retraction. For the backward transformation, points in the cuts are transformed to the empty value ∅. Right: feature-aligned transformation. The extra test cases for backward mapping are shown.

between the discretely sampled points in V'. Regardless whether it is in the image space or the new object space V', it is not possible for a rendering algorithm to determine if it should interpolate between the transformed points. Third, as the transformation is often non-linear, it is difficult to estimate an effective sampling resolution of the final volume in relation to the original volume V. For example, this problem was exhibited in [15] where sampling resolution was set to one sample per voxel, resulting in unintentional discontinuity when stretching any part of the volume. The most effective way to overcome this problem is to employ the inverse transformation T_F^{-1}, provided that the first issue can be correctly resolved.

3.3 Modeling Cuts

Modeling cuts and dissections is not trivial when using a mesh or surface representation. A common approach would be to subdivide a surface mesh to account for the new surface information that arises from the breakage. However, most surface models assume a 0-thickness boundary representation with either empty or homogeneous solid interior, which are not suitable for illustrative manipulation. Complex multi-layer shell representations are therefore necessary, which may lead to further computational costs in determining collision between operators and objects. In addition, the manipulation would not be easily reversible, as combining meshes is another non-trivial problem in surface modeling.

On the other hand, the volume representation does not suffer from the above shortcomings. In volume manipulation, cuts and dissections can be modeled as point-wise mapping and do not involve explicit geometric operations. As mentioned in Section 3.2, the use of inverse transformation T_F^{-1} provides a means for high-quality rendering, but may encounter the problem of undefined transformation in the cut area. Ideally, for any forward transformation, T_F from P_V to P'_V, we should always have its corresponding T_F^{-1}, such that, for each point $p' \in P'_V$, there exists $p \in P_V$ and $p' = T_F(p)$. However, this condition cannot be easily met, because we actually sample the bounding volume V' rather than the unknown point set P'_V, and V' is likely to contain points, for instance, in a cut area, which do not have a pre-image in V.

Let $P'_{V'}$ be a collection of all points in V'. Since P'_V is a set of all points located in V' with a pre-image in V, the empty space in V' is thus defined by a set of points $P'_{empty} = P'_{V'} - P'_V$. We thereby replace the T_F^{-1} in Eq.(1) with a modified *backward transformation* T_B as:

$$p = T_B(p') = \begin{cases} T_F^{-1}(p') & p' \in P'_V \\ \varnothing & p' \in P'_{empty} \end{cases} \tag{2}$$

where ∅ denotes a null position, indicating a point that does not have an origin prior to the manipulation. In general, such points are considered empty, or completely transparent. We thereby assume that, for purposes of rendering, the opacity value associated with ∅ is 0. An example is shown in Figure 4, where the forward and backward transformation of a retractor tool is illustrated. Note that for sampling points in the interior of the cut, the inverse transform is defined as an empty value. In feature-aligned manipulation, T_B needs to be used in conjunction with Eqs.(3) and (4).

4 Feature-Aligned Manipulation

One requirement for illustrative manipulation is the ability to align cuts with features of interest. Therefore it is necessary to have a means for specifying such region. We introduce a *masking function* M, which defines the feature-sensitivity of points in the original volume V, and is typically represented by a volume dataset. When $M(p) \geq 0.5$, p is part of the feature to be preserved, and cannot be transformed. An example of this is shown in Figure 3(c) where the peel is applied to the skin and not to the brains (which is masked as a feature of interest). Let P_M be the subset of P_V, such that $P_M = \{p | p \in P_V, M(p) \geq 0.5\}$, and V_M is an axis-aligned bounding volume of P_M. Any point not in P_M is operatable. In addition, the backward transformation function of each manipulation, T_B, is also accompanied by a *bounding volume*, V_T, such that the manipulation is only performed over those points residing in V_T.

It is possible that the inversely transformed point, p', does not have a pre-image in P_V, or p' has been masked as non-operatable by M. To handle the complexity associated with V_T and V_M, we introduce an initial "probe" p^0, for each point $p' \in P'_{V'}$ as follows:

$$p^0 = \begin{cases} T_B(p') & p' \text{resides in} V_T \\ p' & \text{otherwise} \end{cases} \tag{3}$$

We then obtain p by taking the feature mask into account as:

$$p = \begin{cases} p^0 & p^0 \in P_V \wedge (M(p') < 0.5 \wedge M(p^0) < 0.5) \\ p' & p' \in P_V \wedge M(p') \geq 0.5 \\ \varnothing & \text{otherwise} \end{cases} \tag{4}$$

These three cases are shown in Figure 4 right, namely: (Case 1): the point is transformed, (Case 2): the point is masked and therefore untransformed, and (Case 3): the point is empty due to the feature-aligned cut.

Normal estimation is an essential part of the rendering of graphical objects. Following the same rules as Eq. (4), the normal at a point p' is defined as:

$$\nabla_T(p') = \begin{cases} \nabla_J(p') & p = T_F^{-1}(p') \\ \nabla(p) & p = p' \\ \mathbf{0} & p = \varnothing \end{cases} \tag{5}$$

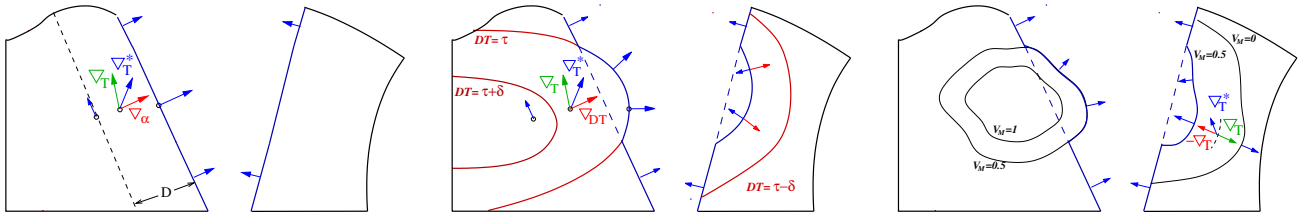
where ∇_T denotes the gradient of the *transformed* point, $\nabla(p)$ denotes the original gradient, ∇_J denotes the gradient at the sampling point p', and $\mathbf{0}$ is the zero vector. ∇_J can be obtained using finite differences on the deformed object. However, this would involve computing the inverse transformation of the adjacent neighbors of p', and would be computationally expensive for real-time rendering. Instead, ∇_J can be estimated at each point via the inverse transpose of the Jacobian of the transformation, as proposed in [1]. The normal transformed via this method is then denoted here with a subscript J.

4.1 Axis Alignment

For axis-aligned manipulation, all points residing in V are operatable, which is equivalent to $P_M = \{\}$. Eq. (4) is thus simplified to:

$$p = \begin{cases} p^0 & p^0 \in P_V \\ \varnothing & \text{otherwise} \end{cases} \tag{6}$$

The normal estimation is also simplified for this case, which reduces to computing ∇_J in most cases. The only challenge here is the computation of the normal in the presence of cuts. Rather than deriving a



(a) Axis Alignment (b) Surface Alignment (based on distance field) (c) Segment Alignment

Fig. 5: Normal blending for the different alignments in the boundaries of cuts. Green arrows indicate the estimated normal from the Jacobian, red arrows are the normal orthogonal to the surface of the cut (propagated along the blending region), and the blue arrows indicate the corrected normal after blending.

complex formula for adjusting the Jacobian in the proximity to cuts, we compute the Jacobian on a continuous version of the transformation, where "holes" are filled with valid points. We then estimate a more accurate normal by adjusting the gradient in the vicinity of cuts, by blending from the transformed normal ∇_T, to the "expected" normal from the surface of the cut, which we denote as ∇_α. The blending function can then be defined as:

$$\nabla_T^*(p') = (1-\omega)\nabla_T(p') + \omega\nabla_\alpha(p') \tag{7}$$

where $\omega \in [0,1]$ is a weighting function that decreases with the distance to the discontinuity, i.e., for a point in the boundary, $\omega = 1$, and for a point at a pre-defined distance D from the boundary, $\omega = 0$. This parameter ω controls the gradient smoothness of the cut surface, and it is similar to the "impregnation" region described by Weiskopf et al. in [23] for performing volumetric clipping. The blending mechanism is depicted in Figure 5(a).

Standard manipulation operators can be placed anywhere within the volume by allowing T_B and its bounding volume V_T to align with an arbitrary axis plane. The general mechanism for performing such alignment is to transform the coordinate frame of the operator before applying it to a volume. As an inverse warping problem, this is equivalent to applying the inverse coordinate transformation to the sampled points p', that is, referring to Eq. (2) we have:

$$p = \mathsf{A} \times \left(T_B(\mathsf{A}^{-1} \times p')\right) \tag{8}$$

where A is a 4×4 affine matrix, which is constant for all the points in the sampled volume. Normals can be obtained by concatenating the Jacobians of the affine transformations with the Jacobian of the manipulation operator, which only requires a multiplication by two constant matrices. Manipulating operators is supported by this mechanism for arbitrary axis alignment.

4.2 Surface Alignment

As stated previously, for certain manipulations axis alignment is not sufficient. Peeling is one such manipulation, since generally the peeling operation is applied to a specific surface layer of an object. In this case, feature-based alignment is desired. If features are not segmented or preassigned, an approximation can be obtained by defining a mask based on distance from the surface. This can be done with the distance field of the volume, after a background segmentation. Let us define DT as the distance field stored as a volume. Then, the mask M can be defined such that $M(p) \geq 0.5$ for $DT(p) \geq \tau$, and $M(p) < 0.5$, for $DT(p) < \tau$. Here $\tau > 0$ is a parameter that specifies the desired distance from surface. For instance, τ can be thought of as the "depth" of the peeled surface, which is to be transformed in order to reveal the feature underneath.

Estimation of the normals requires special handling of the boundary around the cut surface defined by $DT(p) = \tau$. For a feature point (i.e., $M(p) \geq 0.5$) the gradient of the distance field ∇_{DT}, points outwards from the interior to the surface. However, for a non-feature point on the boundary (i.e., $M(p) < 0.5$), the gradient ∇_{DT} points incorrectly from the surface to the interior. We solve this by using a signed weighting function β. We thus have:

$$\begin{aligned} \nabla_T^*(p') &= (1-|\beta|)\nabla_T(p') + \beta\nabla_{DT}(p') \\ \beta &= \begin{cases} \frac{\tau - DT(p)}{\delta} - 1 & \tau - \delta < DT(p) < \tau \\ \frac{\tau - DT(p)}{\delta} + 1 & \tau \leq DT(p) < \tau + \delta \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

β is used to gradually blend the normal of the distance field with the transformed normal. Note that it is asymmetric with respect to τ. That is, for a point with DT in the interval $[\tau, \tau+\delta)$, β ranges from 1 down to 0, but for a point with DT in the interval $(\tau-\delta, \tau)$, β takes values from 0 down to -1. This negative weighting blends the transformed normal ∇_T and the normal between the normal of the distance field in the *opposite* direction. This results in correct normals at both sides of the break. This is illustrated in Figure 5(b), and an example of surface-aligned peeling is shown in 3(b).

4.3 Segment Alignment

Segment alignment is obtained by defining $M(p)$ based on a volume feature, that typically is determined through segmentation. It can be seen quite easily that the above technique for surface alignment can be extended to handle an arbitrary volumetric mask by replacing $DT(p)$ directly with $M(p)$. Normal adjustment is handled as follows:

First, we assume that the gradient of the original volume is computed with the aid of the segmentation information, stored as a volume texture. This correctly estimates the surface gradient of the features of interest. Further, for the boundary between two features, the normals on either side point to the opposite direction of those of the other side. This leads to a problem in texture-based volume rendering, since trilinear interpolation of these opposite normals would yield an incorrect zero gradient. To overcome this problem, we estimate the gradients for segmented data so that they always point outwards from the feature of interest on both sides of the boundary. When computing the gradient volume texture using finite differences, we consider the values of the neighbors of a voxel as 0, if they correspond to a different segment or an empty voxel, or as 1 if they correspond to the segment of interest.

Finally, we invert the normals in the non-feature side of the cut following the blending mechanism in Section 4.2. To define a thick area where this blending can be possible, we assume that the mask $M(p)$ defines a smooth scalar field. The region where we need to invert the directions of the normals is defined by the isosurfaces $M(p) = 0.5$ and $M(p) = 0$, as shown in Figure 5(c). This results in the following equation:

$$\begin{aligned} \nabla_T^*(p') &= (1-2\gamma)\nabla_T(p') \\ \gamma &= \begin{cases} 2M(p) & 0 < M(p) < 0.5 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{10}$$

Fig. 6: Segment alignment (left) vs. Two-pass rendering on two pre-segmented datasets (middle). Note how the bone segment is properly classified on the left. The insets on the right highlight the aliasing effect of the two-pass rendering (bottom), while there is no noticeable aliasing with the segment alignment (top).

Previous approaches for achieving the same results of segment alignment often use pre-segmented datasets and two-pass rendering, where the segment of interest to be preserved is rendered first, and then the non-feature part of the volume is manipulated and rendered afterwards. However, this approach results in aliased boundaries, because of pre-segmentation, and in incorrect post-classification of boundary voxels. Figure 6 compares our method with this alternative. Note that on the left, segment alignment properly classifies the bone voxels, while the two-pass rendering (middle) gives a tint similar to the color of the skin. On the right, the two insets highlight the different levels of aliasing on the bone segment. Although these problems can be addressed with pre-smoothing and pre-processing of colormaps, our approach works on a single pass and requires no pre-processing of the dataset or the colormap.

5 GPU Implementation

For real-time manipulation, a GPU implementation has been developed. It is based on texture-based view-aligned slice volume rendering [24]. We assume that the original volumetric object is stored as a 3D texture. In traditional volume rendering, the view-aligned slices are used to sample the 3D texture at regular intervals. In our approach, we sample the *deformed* space, i.e., we will be performing inverse mapping. The feature mask M is also stored as a 3D texture. There are a number of ways of defining the transformation T_B of an operator: explicitly in the fragment shader, which is usually computationally expensive, or sampled in a volumetric domain and stored in a 3D texture. The latter involves the pre-computation of the transformation *a priori*, and storage on a texture of the result of the operation, as described in [2] or a 3D displacement, as described in [6]. We chose 3D displacements because of its flexibility and generality. The union of all the operators and the original volume creates V'. This is then sliced along the view direction. As each slice is rendered, a fragment is generated for each pixel in the image. Thus, each fragment generated by rendering a single slice corresponds to a point in the deformed volume. First, the corresponding point in the undeformed volume is found by evaluating Eq. (4). Then, the original volume is sampled to find the voxel density values. In Eq. (4), voxel positions defined as $\varnothing$ should not contribute to compositing, so they are defined as having opacity zero. To obtain the correct color attributes, the normals must be determined, by evaluating Eq. (5). The normal of each fragment requires at most three gradient texture samples: the normal obtained from the transformation (∇_T), as discussed in Section 4.1, the normal of the cut (∇_α), and the normal of the feature mask, depending on the alignment. These are blended together as shown in Eqs. (7), (9) and (10). The gradients can also be stored as textures to speed up computation.

The GPU memory requirements for this process are predominantly determined by the resolution needed to store the volumetric dataset (e.g., the head dataset requires 256^3 bytes and its gradient requires 3×256^3 bytes). An additional requirement is imposed by the pre-computation of the manipulation operators. However, these are in general very small compared to the 3D volume data. When resources are scarce, normals can be computed on the fly using finite differences, not only for the original dataset, but also for the alignment mask and the operator itself.

5.1 Multiple Manipulations

We have considered the case of a single manipulation applied to a volumetric model. However, as seen in Figure 1(c), it is possible to apply several operators simultaneously to a single dataset. We consider here two distinct cases for introducing multiple manipulations. In the first case, each operator represents an independent manipulation on the dataset, and the result of one does not affect the other. We treat each operator as its own proxy geometry, and the original dataset as a "background" proxy geometry. When rendering the scene, the bounding box for each operator is sampled independently with the appropriate manipulation parameters. Finally, the scene is composed with the slices resulting from all the intervening operators. However, overlapping with the background volume may occur. A stencil is used to mask out the operator regions so they are not rendered twice. An example is shown in Figure 1(c).

When manipulations overlap, this case is more complicated, since it requires us to define an algebraic operation for the combination of two operators. There are a number of ways to deal with this situation. For example, if the dataset is segmented, each "segment" (i.e., skin vs. organs) is sliced independently, and the resulting slices are composited into a single image. This can be seen in Figure 10(d) where the retractor operator is used to open the skin of the frog, while a plier operator is used to poke and pull the internal organs. Another way is to save the intermediate volume into a new dataset and used the saved volume to apply other operations. These approaches are under investigation.

6 Applications

One of the applications of feature-aligned volume manipulation is medical and biological illustrations. In Figure 1(b), an illustration from hand surgery is shown. Figure 1(a) demonstrates a retraction operator on a CT hand mimicking the same type of cut. Figure 1(d) is an image from the illustration of human anatomy by Antonio Scrantoni and Paolo Mascagni, dated 1833 (NLM). Interestingly, this image is very similar to contemporary exhibitions such as BodyWorlds [22] and Bodies, The Exhibition [9] which portray dissections of actual bodies. Figure 1(c) shows a similar type of operation applied to the Visible Man dataset. The dataset was first posed using [10] to position the arm upright. Five peel operators were then applied to both arms.

Figure 7 shows a volume manipulation similar to illustrations of forefoot surgery using the retractor operator. Figures 7(a) and (b) show the result of surface-aligned manipulation, for a distance parameter of $\tau = 0.08$ and $\tau = 0.026$, respectively ($\tau = 0$ corresponds to the surface of the dataset). Figure 7(c) shows the result of segment-aligned manipulation to obtain a better visualization of the bone tissue. 7(d) shows a close-up view of another cut, where it is possible to see different features by applying different layer levels. In the upper image, only bones are visible, whereas in the lower portion, we can see the vessels and muscle tissue below the skin. Figure 8 shows the dilation operator applied to a colon dataset. This is an affine alignment along a portion of the colon. The bounding box of the operator, is shown, so too part of the bounding box of the original volume V. Figure 9(a) shows a retraction around the torso which moves tissue and reveals internal organs. In Figure 9(b) the plier operator pushes an occluding artery to reveal the spine. Figure 10 shows an application of feature-aligned manipulation, where a retractor operator is applied to the segmented frog dataset. In order to do this, we define the manipulation mask as the skin and muscles, so the internal organs remain untransformed.

Alignment	Dataset	Resolution	fps
Axis	foot	$143 \times 256 \times 183$	20.44
	stag beetle	$208 \times 208 \times 123$	9.24
	visible human 2mm	$256 \times 189 \times 436$	6.27
	cthead	$256 \times 256 \times 256$	5.06
Surface	foot	$143 \times 256 \times 183$	18.31
	stag beetle	$208 \times 208 \times 123$	7.32
	orange	$256 \times 256 \times 256$	6.11
	cthead	$256 \times 256 \times 256$	3.93
Segment	foot	$143 \times 256 \times 183$	17.95
	hand	$255 \times 250 \times 155$	5.54
	frog	$250 \times 235 \times 68$	8.08

Table 1: Performance results for different volume datasets, with $d = 1.0$ for the distance between view aligned slices of the volume

The last image shows a plier operator deforming one of the organs inside the frog. Figure 11 shows an illustration of the peeling of the stag beetle dataset, where a surface aligned peeling is used to remove only the outer layers. Figure 12 shows a peeled orange. The peel is defined using surface alignment.

In addition to illustration-like effects, the manipulation operators also improve on clipping and slicing and generate focus+context visualizations. Now slices can be arbitrary geometries, and there is a focus+context mechanism (peeling) for keeping the sliced portion in view. In Figure 3(c) one gets to see the underside of the peel or skin. Other examples can be seen on the accompanying video, and online at: `http://www.caip.rutgers.edu/~cdcorrea/feature`.

7 Performance Evaluation

Since we follow a slicing approach to render our scenes, rendering performance is mainly influenced by the fragment shader capabilities of the graphics board. The rendering speed is affected by a number of factors, including: sampling distance of the slices, resolution of the object, viewport size, and the number and the relative size of the operators with respect to the volume. We implemented the manipulation operators within an interactive program which allows the user to rotate and scale the object, to move or change the operator, and to change the color map or lighting parameters.

Our test configuration consists of a Pentium XEON 2.8 Ghz PC with 4096 MB RAM, equipped with a Quadro FX 4400 (512MB). We tested our approach on a number of datasets, ranging in size from 128^3 to 256^3, in a 512×512 viewport. Table 1 shows the results obtained with our test datasets for the different alignment cases and a slicing distance of 1.0. For higher quality rendering, a distance of $d = 0.5$ can be used. In this case, the rendering rate was found to be exactly one half of the rate with $d = 1.0$.

8 Conclusions

We presented a general mechanism for interactive manipulation of volumetric models, which enables the creation of illustrative visualization with complex deformation involving cuts, dissections and other forms of manipulations. We described a new approach that allows the manipulation operators to be aligned to the features of interest. With both surface and segment-based features, more complex manipulation effects can be realized, representing a further generalization of the traditional axis-oriented volume manipulation technique. We presented several mechanisms for estimating accurately the normals in the deformed volume space and described a feature-sensitive mechanism for adjusting normals in the vicinity of cuts. We provided a GPU-based implementation that renders illustrative manipulation in real time with a quality that is comparable with that obtained using the non-interactive raycasting method. Future research involves efficiently implementing overlapping manipulations, investigating user interface and usability issues, using these techniques for virtual reality applications, and the inclusion of other operators. Through a number of examples, inspired by medical and biological illustrations, we have shown that this technique provides the necessary interactivity and operatability in the process of creating various manipulation effects in illustrative visualization.

Acknowledgements

Volumetric datasets are courtesy of Lawrence Berkeley Laboratory, UNC Chapel Hill, University of Iowa, U.S. National Library of Medicine, Viatronix Inc. and Viena University of Technology. We want to thank Dr. Stanley Trooksin, Dr. Sid Roychowdhury and Dr. Marsha Jessup for valuable input on surgical and medical illustration. The illustrations in the paper are courtesy of medmovie.com and Nucleus Medical Art, Inc.

References

[1] A. Barr. Ray tracing deformed surfaces. *Computer Graphics (Proc. SIGGRAPH 86)*, 20(4):287–296, 1986.

[2] T. Brunet, K. Nowak, and M. Gleicher. Integrating dynamic deformations into interactive volume visualization. In *Eurographics /IEEE VGTC Symposium on Visualization 2006*, pages 219–226, 2006.

[3] H. Chen, J. Hesser, and R. Männer. Ray casting free-form deformed-volume objects. *J. of Vis. and Computer Animation*, 14(2):61–72, 2003.

[4] M. Chen, C. Correa, S. Islam, M. W. Jones, P.-Y. Shen, D. Silver, S. J. Walton, and P. J. Willis. Deforming and animating discretely sampled object representations. In *Eurographics State of the Art Report*, 2005.

[5] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proc. Volume Graphics '03*, pages 35–44, 2003.

[6] C. Correa, D. Silver, and M. Chen. Discontinuous displacement mapping for volume graphics. In *Proc. Volume Graphics '06*, 2006.

[7] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. *Comput. Graph. Forum*, 22(3):523–532, 2003.

[8] D. Ebert and P. Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *IEEE Visualization*, pages 195–202, 2000.

[9] B. T. Exhibition, 2005.

[10] N. Gagvani and D. Silver. Animating volumetric models. *Graphical Models*, 63(6):443–458, 2001.

[11] S. Gibson and Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR97-19, MERL Technical Report, 1997.

[12] S. Islam, D. Silver, and M.Chen. Volume splitting and its applications. *IEEE Transactions on Visualization and Computer Graphics*, To appear.

[13] Y. Kurzion and R. Yagel. Interactive space deformation with hardware-assisted rendering. *IEEE Comput. Graph. Appl.*, 17(5):66–77, 1997.

[14] E. C. LaMar, B. Hamann, and K. I. Joy. A magnification lens for interactive volume visualization. In *Pacific Graphics 2001*, pages 223–232, 2001.

[15] M. J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *IEEE Visualization 2003*, pages 401–408, 2003.

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

[17] M. Peercy, J. Airy, and B. Cabral. Efficient bump mapping hardware. In *Computer Graphics, Proc. SIGGRAPH '97*, pages 303–307, 1997.

[18] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume rendering on standard PC graphics hardware using multi-textures and multi-state rasterization. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 109–118, 2000.

[19] C. Rezk-Salama, M. Scheuering, G. Soza, and G. Greiner. Fast volumetric deformation on general purpose hardware. In *Proc. SIGGRAPH/Eurographics Graphics Hardware Workshop 2001*, pages 17–24, 2001.

[20] V. Singh, D. silver, and N.Cornea. Real time volume manipulation. In *Proc. Volume Graphics 2003*, pages 45–51, 2003.

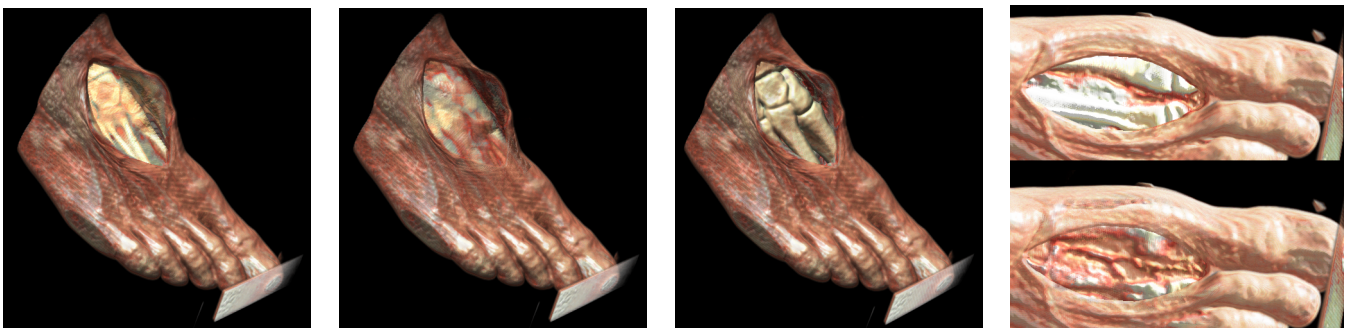
[21] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A simple and flexible volume rendering framework for graphics-hardware-based ray casting. In *Proc. Volume Graphics 2005*, pages 187–195, 2005.

[22] G. von Hagens' Bodyworlds, 2005.

[23] D. Weiskopf, K. Engel, and T. Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Trans. Vis. Comput. Graph.*, 9(3):298–312, 2003.

[24] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. In *Computer Graphics (Proc. SIGGRAPH 98)*, pages 279–296, 1998.

[25] R. Westermann and C. Salama. Real-time volume deformations. *Computer Graphics Forum*, 20(3), 2001.



(a) (b) (c) (d)

Fig. 7: Retractor operator applied to a CT foot dataset. (a-b) Surface alignment with two different layer depths, one revealing bone, and the other reveals superficial veins. (c) Segment alignment showing bone tissue. (d) Zoomed-in surface aligned cut.

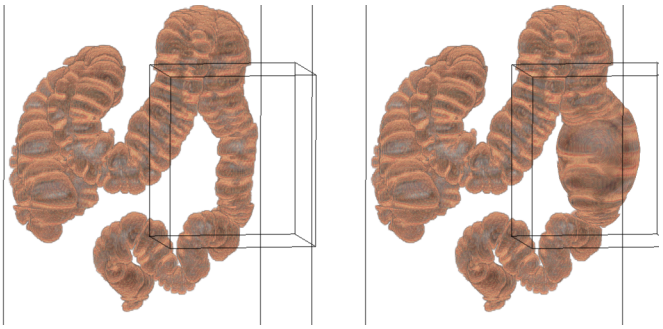


Fig. 8: Dilation operator applied to a colon dataset. (a) Without dilation (b) With dilation. Bounding area of dilation is shown. Dilation works as a type of volumetric lens.

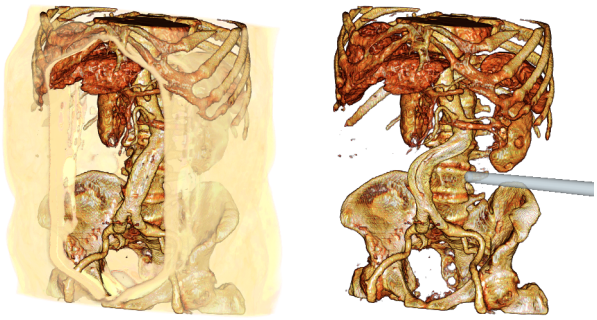
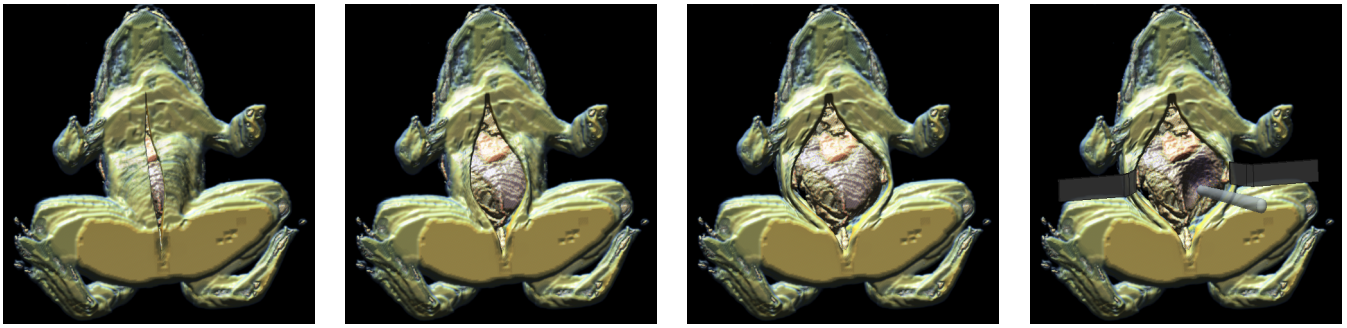


Fig. 9: Manipulation operators used to improve visibility on a CT dataset. The spreader (a type of retractor) moves tissue and reveals internal organs. (b) The pliers push an occluding artery to reveal the spine. This works as a focus+context visualization.



(a) (b) (c) (d)

Fig. 10: (a-c) Retractor operator used to simulate dissection of a segmented frog dataset. (d) A plier operator is applied to the internal organs, while simultaneously retracting the skin. Geometric models are embedded in the scene to show the placement of the operators.

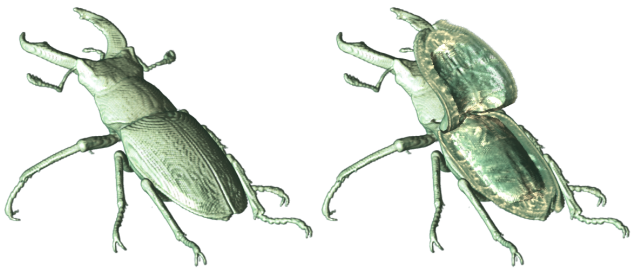


Fig. 11: Surface-aligned peeling of the stag beetle dataset. Note how the internal features can be seen by lifting the wing shells.

Fig. 12: Surface-aligned peeling of an orange. The internal structure of the orange is clearly visible without the need of segmentation. Different placement of the operator shows two stages of an orange peel