Volume Graphics (2007)
H. - C. Hege, R. Machiraju (Editors)

Volume Deformation via Scattered Data Interpolation

Carlos D. Correa[1] Deborah Silver[1] Min Chen[2]

[1]Rutgers, The State University of New Jersey [2]University of Wales Swansea, UK

Abstract

With the advent of contemporary GPUs, it has been possible to perform volume deformation at interactive rates. In particular, it has been shown that deformation can be important for the purposes of illustration. In such cases, rather than being the result of a physically-based simulation, volume deformation is often goal-oriented and user-guided. For this purpose, it is important to provide the user with tools for directly specifying a deformation interactively and refine it based on constraints or user intention. In many cases, deformation is obtained based on a reference object or image. In this paper, we present a method for deforming volumetric objects based on user guided scattered data interpolation. A GPU-based implementation enables real-time manipulation of 2D images and volumes. We show how this approach can have applications in scientific illustration, volume exploration and visualization, generation of animations and special effects, among others.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

1. Introduction

Recent advances in GPU technology has enabled the interactive manipulation and deformation of volumetric objects, including those obtained from CT or MR images, procedurally defined objects, or as the product of a simulation. Because of the complexity of volumetric objects, specifying deformations is a complex task. Surface-based deformation methods are often of limited use in volume graphics due to the inability to represent the desired deformation of the interior of objects, and because volumetric data often do not contain a explicit representation of the geometry of the objects. Direct transformation of voxels is impractical and often results in undesired aliasing and undersampling effects [MTB03]. In other cases, volume deformation is reduced to the direct manipulation of the vertices of a control mesh, such as a free-form lattice [WRS01], which limits the types of deformations that can be attained. Discontinuous displacement maps overcomes this limitation by allowing arbitrary deformations and cuts to be applied to volumetric objects [CSC06b], at the cost of pre-computing displacement maps, which can be difficult to create and place in 3D space to obtain a desired deformation. Other methods represent deformation as the product of physics-based simulation, but they are computationally costly, and difficult to control when the user wishes to attain a "goal" deformed object.

For this reason, it is important to provide a method for volume deformation that decreases the complexity of manipulating millions of voxels in 3D space to simpler and fewer elements. In this paper, we present a method for specifying the deformation of a 3D volumetric object via scattered data interpolation. In our approach, we guide the 3D deformation starting from a user defined illustration. The user deforms this 2D image by transforming a set of control points. Scattered data interpolation is then used to build a 2D displacement map, which is "extruded" into a 3D displacement map according to simple user-defined rules. This displacement map is then applied to the object to obtain a deformed volume, using the methodology in [CSC06b, CSC06a].

This method is advantageous over previous approaches, as the complexity of handling a large number of control points in 3D space is reduced to a 2D image. By using scattered data interpolation, it is possible to build a smooth displacement map that results in plausible deformations. Although the use of 2D images limit the types of deformation that can be generated in 3D, it is possible to extend this idea to direct 3D deformation. We believe this has applicability in 2D

and 3D image morphing and animation, where deformation is guided by a set of matching feature points, in illustration, where deformation is often guided by 2D reference images, and in volume graphics in general.

2. Related Work

Volume deformation has been possible with the advent of programmable GPU's. One of the first to propose this was Westermann and Rezk-Salama [WRS01, RSSSG01], who define deformation as the transformation of nodes located in a proxy mesh, placed in the boundary of the volume. Because of the existence of an embedding geometry, goal-driven deformations are difficult to obtain as control points are not placed within features in the volumetric object. The use of matching features for deformation has only been proposed before for volume morphing, where an initial and end volumes are required [FHSR96, LGL95]. These, however, were intended for non interactive generation of intermediate volumes, rather than for interactive deformation. Other volume deformation approaches use pre-computed procedures to define the transformation. These include spatial transfer functions [CSW*03], ray deflectors [KY97], and discontinuous displacement maps [CSC06b, CSC06a]. Although these solve the problem of expressing complex deformations, the creation and placement in 3D of these tools is a major problem. In this paper, we present a method for obtaining 3D displacements using scattered data interpolation. A smooth displacement map is obtained by interpolating the displacement of a small number of control points, manipulated directly by the user. We expand on the work in [CSC06b], by allowing the user to create the desired deformation through an interactive illustration. Scattered data interpolation has been used for the deformation of 2D images. In [RM95], a number of approaches are explored, including inverse distance weighted interpolation and radial basis functions (RBFs). RBFs including rigid structures was explored by [LHH97]. Recent approaches to image deformation are aimed to finding as-rigid-as-possible deformations to be effective [ACOL00, SMW06]. In our approach, we use rigidity constraints with Radial Basis Functions to provide more plausible deformations. Unlike these previous approaches, we do not require a triangular mesh of the image, but rather work directly on a per-pixel basis. Further, we propose an extension which allows the introduction of discontinuities, useful for depicting cuts. The idea of scattered data interpolation was also introduced to the deformation of 3D objects [RM93, KSSH02, BK05].

3. Overview

Our approach works in two stages. In the first stage, the user creates the desired deformation by illustrating on a 2D image. The 2D image is obtained from the volumetric object, either as a composited projection, a maximum intensity projection (MIP) or directly from a data slice. This deformation is performed by defining a set of control points and transforming them directly. The image is deformed by interpolating the displacement of the control points along the entire surface of the image. Once a desired displacement is obtained, a 3D displacement is computed, by extruding the 2D displacement along the z direction, using simple rules defined by the user. One such rule could be simply to replicate the 2D displacement into a 3D volume. Then, the 3D displacement is used to deform the volume using the inverse warping methods proposed in [CSC06b]. The following sections describe this process in more detail.

3.1. Deformation via Scattered Data Interpolation

In the first stage of the process, we present the user with a 2D projection of the 3D volume. This projection can be a representative slice, or a maximum intensity projection (MIP) obtained from the user's current viewpoint.

First, we assume we have a set of control points in R^2, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, and we define an inverse displacement for each point, that is, we have a set of displacements in R^2, $\{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N\}$. A displacement map can be found by finding a continuous function that interpolates these displacement values at the control points. A number of approaches have been proposed, such as inverse distance weighted interpolation, where the interpolant is built as weighted average of the displacement. Another method is via Radial Basis Functions (RBF), where the interpolant is a linear combination of basis functions, which only depend on the distance to the control points (thus named radial). In this work, we use RBF to construct our interpolant.

Given a set of points X and a set of values $f_1, \ldots, f_N$, a continuous function f can be found with the RBF interpolant:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi(||\mathbf{x} - \mathbf{x}_i||) + p_k(\mathbf{x}) \qquad (1)$$

where α_i are a set of coefficients and $p_k(\mathbf{x})$ is a polynomial of degree k. $\phi(\cdot)$ is a radial basis function, i.e., it only depends on the euclidean distance to a point (here denoted $||\cdot||$). The coefficients α_i and of the polynomial are found by putting the control points into the interpolant, i.e., enforcing $f(\mathbf{x}_i) = f_i$.

For deformation, we want to obtain a 2D function $\mathbf{D}$, corresponding to the displacement in the x and y directions. Then the interpolant is defined as:

$$\mathbf{D}(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{a}_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \mathbf{p}_k(\mathbf{x}) \qquad (2)$$

where $\mathbf{a}_i = [\alpha_i, \beta_i]^\top$ is a tuple of coefficients and $\mathbf{p}_k = [p_k, q_k]^\top$ is a tuple consisting of two polynomials of degree k. It is common practice to consider the 2D function as two independent functions d_x and d_y, and use them inde-

pendently to interpolate the x and y components of the displacement, respectively.

In our work, we use polynomials of degree $k = 1$, which ensures that the deformation would be as-affine-as-possible. Then, the polynomials are of the form $p_k(\mathbf{x}) = \gamma_0 x + \gamma_1 y + \gamma_2$ and $q_k(\mathbf{x}) = \lambda_0 x + \lambda_1 y + \lambda_2$.

In order to find the coefficients, we put the control values into Eq.(2) and solve the resulting linear system. For the x and y components of the displacement, it leads to two symmetric linear systems:

$$\begin{pmatrix} \Phi & P \\ P^\top & 0 \end{pmatrix} \begin{pmatrix} \alpha_j \\ \gamma_l \end{pmatrix} = \begin{pmatrix} dx_i \\ 0 \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} \Phi & P \\ P^\top & 0 \end{pmatrix} \begin{pmatrix} \beta_j \\ \lambda_l \end{pmatrix} = \begin{pmatrix} dy_i \\ 0 \end{pmatrix} \tag{4}$$

where Φ is a matrix of size $N \times N$, such that $\phi_{ij} = \phi(||\mathbf{x}_i - \mathbf{x}_j||)$, P is a matrix of size $N \times 3$ where $P_i = (1, x_i, y_i)$ and dx and dy are the displacements in x and y respectively for the control points.

This system can be solved with a variety of methods, including iterative approaches. In this work, we consider a small number of control points, which are sufficient to sketch a wide range of complex deformations, so that the system can be solved exactly or via least squares estimation at interactive rates.

Radial Basis Functions. A number of radial basis functions $\phi(r)$ have been proposed for scattered data interpolation, with different computational cost and continuity guarantees [RM95]. They are called radial because they only depend on the distance to the control points. In this work, we chose Wendland's polynomial $\psi_{3,1}(r) = (1-r)_+^4(4r+1)$, where $(1-r)_+^4 = (1-r)^4$ for $0 \leq r \leq 1$ and 0 otherwise, for its compact support and because it provides C^2 continuity [Wen95]. Compactly supported RBFs are desired especially when the number of control points increases, since otherwise the interpolant has to be computed based on all points.

3.2. Generation of 3D Displacements

In order to use this method for volumetric deformation, we generate a 3D displacement from the 2D displacement. First, the 2D displacement is obtained by evaluating Eq.(2) at discrete positions in a 2D grid. Then, the 2D displacement is extruded in an orthogonal direction to create a 3D displacement. This extrusion is a 1D interpolation based on user-defined rules. Examples of these rules are:

Replication. In the simplest case, the 3D displacement is obtained by simply replicating the 2D displacement along the orthogonal direction (which we will assume it to be along the z-axis). Then, the 3D displacement is obtained as: $\mathbf{D}_{3D}(x,y,z) = \mathbf{D}(x,y)$ where $\mathbf{D}(x,y)$ is obtained from Eq.(2).

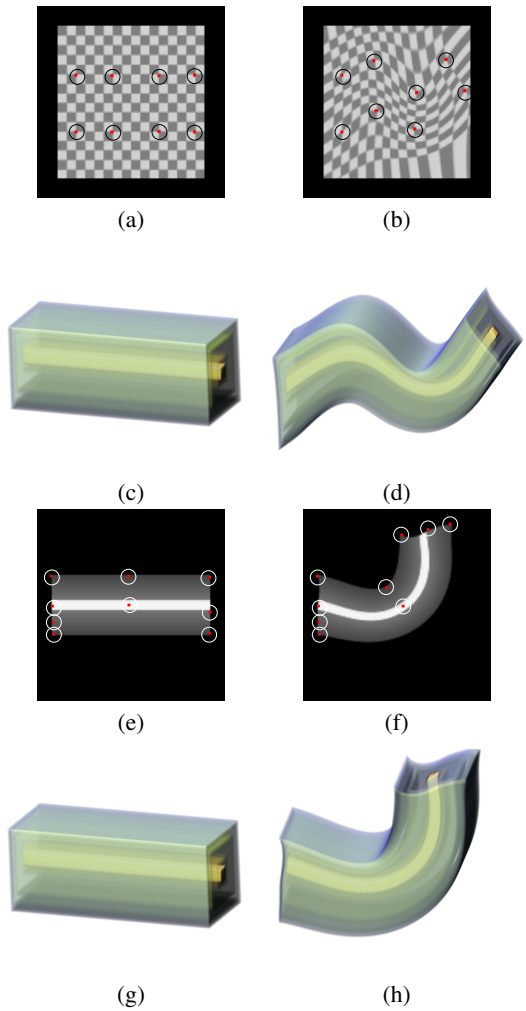


Figure 1: *Deformation of a volumetric bar (created procedurally) (a),(b) Nodal deformation on a 2D grid. Control points are marked with circles. (c),(d) Resulting deformation on the 3D dataset. (e),(f) 2D deformation using a data slice. (g),(h) Corresponding 3D deformation.*

This is the approach taken in Fig. 1, where the 2D deformation is replicated along the z-direction, to represent a bending deformation.

Modulation. In this case, the 3D displacement is obtained by modulating a function of the z coordinate with the 2D displacement. This method is useful for progressively applying the deformation or to provide a smooth deformation from the boundaries of the 3D displacement. In general, the 3D displacement is obtained as: $\mathbf{D}_{3D}(x,y,z) = f(z)\mathbf{D}(x,y)$ When $f(z) = z$, for example, the 3D displacement is smoothly interpolated from zero to the 2D displacement, giving a smooth transition. An example can be seen in Fig. 2(d), where the deformation is smoothly interpolated in the z direction. Compare to Fig. 2(c), which uses the replication rule.

Interpolation. In the most complex case, 3D displacement is obtained by interpolation of two or more 2D displacements. In this case, tri-linear interpolation provided by the graphics hardware can be exploited. One of the applications of this method is the creation of 3D displacements that can be applied progressively to a volumetric object to represent different stages of a procedure, e.g., a bending operation. In this case, we keep intermediate 2D displacements obtained as the user moves the control points and use them all together to create a 3D displacement.

Once a 3D displacement is created, it can be applied to the rendering of deformed volumetric objects using the approaches defined in [CSC06b,CSC06a], where inverse warping is used to transform each sample point via the 3D displacement.

4. Complex Deformations

One of the advantages of using Radial Basis Functions as basis for creating an interpolant is the ability to build displacements with certain degree of continuity, and the ability to represent as-affine-as-possible transformations due to its optimization mechanism. However, certain deformations may be difficult to represent due to this. For instance, in many cases, it is desired to have as-rigid-as-possible deformations, rather than affine. In other cases, such as when simulating cuts, global continuity may be a limiting aspect. Here, we describe some ways in which our approach can handle these cases.

4.1. Simulating Rigid Transformations

One of the limitations of the above formulation of the interpolant is that it tries to produce affine transformations whenever possible, due to the definition of the polynomial $\mathbf{p}_k$. Although this is acceptable in many cases, deformations that include rotations, such as a bend, would result in undesired effects. For this reason, we can include additional constraints to the problem, so that a more rigid transformation is obtained. Because we are using a linear polynomial in the interpolant, the interpolation tries to deform points via the transformation:

$$\begin{aligned}\mathbf{x}' &= M\mathbf{x}+T \\ &= \begin{pmatrix} \gamma_0 & \gamma_1 \\ \lambda_0 & \lambda_1 \end{pmatrix}\mathbf{x}+\begin{pmatrix} \gamma_2 \\ \lambda_2 \end{pmatrix}\end{aligned}$$

where γ_i and λ_i are the polynomial coefficients for the x and y components of the interpolant. To ensure a similarity transformation, instead of affine, we must ensure that $M^\top M = \rho^2 I$, where ρ is a uniform scaling and I is the identity matrix. For a rigid transformation, $M^\top M = I$. However, these are non-linear constraints, which cannot be incorporated into the linear formulation. Instead of finding an exact solution to this, we linearize the constraints. For the case of similarity, this can be achieved by adding the constraints $\gamma_0 = \lambda_1$ and $\gamma_1 = -\lambda_0$. This new problem then requires finding the coefficients in a single linear system, rather than two independent systems for each component of the displacement. The new system is then:

$$\begin{pmatrix} \Phi & 0 & P & 0 \\ 0 & \Phi & 0 & P \\ P^\top & 0 & 0 & 0 \\ 0 & P^\top & 0 & 0 \\ 0 & 0 & \kappa_1 & -\kappa_2 \\ 0 & 0 & \kappa_2 & \kappa_1 \end{pmatrix}\begin{pmatrix} \alpha_j \\ \beta_j \\ \gamma_l \\ \lambda_l \end{pmatrix} = \begin{pmatrix} dx_i \\ dy_i \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (5)$$

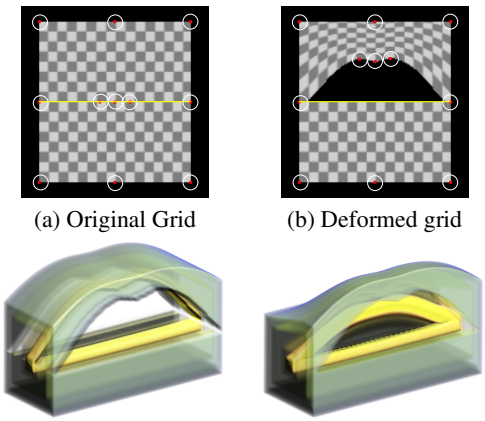
where $\kappa_1 = [1,0,0]$ and $\kappa_2 = [0,1,0]$ are the linearized constraints for the similarity transformation. This is a different approach than the one in [SMW06] for enforcing constraints, but our results seem very promising. An example is shown in Fig. 4, where a knee is bent using a single control point in the lower part of the leg. Note the difference between the deformation in Fig. 4(c), where rigidity constraints are added and that in Fig. 4(d), with no constraints. The latter results in unrealistic squashing of the leg, while the former preserves for most part the rigidity of the bent bones, and the rest is deformed smoothly. This approach, although very fast, has a limitation: rigidity constraints are only loosely enforced due to linearization, and the resulting deformation contains a uniform scaling factor. Following the approach in [IMH05] one can regularize the deformation by adjusting the scaling after solving the RBF equations. Another approach to introduce rigidity constraints is borrowed from the non-linear medical image registration community [LHH97]. In this case, rigidity constraints are included by considering a set of "landmarks", each of which is moved rigidly. These landmarks are useful not only to include rigid features, but also to fix certain parts of the dataset. Following the approach in [LHH97], we can introduce a rigid part by modifying the Radial Basis Function $\phi(r)$, such that it approaches zero as a point gets closer to the rigid feature. This can be achieved by considering the distance transform of the rigid part. Then, a new RBF can be built for a pair of points $\mathbf{x}$ and $\mathbf{x}_i$

$$\phi_R(\mathbf{x},\mathbf{x}_i) = \mathscr{D}(\mathbf{x})\mathscr{D}(\mathbf{x}_i)\phi(||\mathbf{x}-\mathbf{x}_i||) \quad (6)$$

where $\mathscr{D}(\mathbf{x})$ is the unsigned distance transform of the rigid part, so that it is zero for points inside it. This ensures that all points within a rigid part have an RBF of zero. An example is shown in Fig. 6(c),(d), where it is applied to a volumetric representation of a horse model. The rigidity constraints are used to fix the legs of the horse, as a distance transform of the horizontal plane where the horse stands. The incorporation of more complex rigidity constraints is currently being sought.

4.2. Simulating Cuts

One of the advantages of RBFs is the ability to obtain displacements that are at least C^1 continuous. However, this poses a problem for defining discontinuous deformation. In the case of cuts, C^1 continuity is desired in the sub-regions



(a) Original Grid (b) Deformed grid

(c) Generated Cut (d) Modulated Cut

Figure 2: *Deformation cut added by the user using a 2D grid as a guide (a,b). The 3D displacement is found using Decoupled Radial Basis Functions, and extruding along the Z direction via replication (c) and modulation (d).*

that appear after the cut. For this reason, we devised what we call *Decoupled Radial Basis Functions* (DRBF). For simplicity, let us consider a DRBF in 2D with two continuity regions (i.e., there is a single cut). The two continuity regions divide the image into the regions I_P and I_Q. Therefore, $I_P(\mathbf{x}) = 1$ if $\mathbf{x}$ is in I_P and 0, otherwise, and similarly for I_Q. In addition, we assign control points to a particular region, i.e., control points are divided into the sets $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$, respectively. The 2D displacement with cuts is then given by:

$$D(\mathbf{x}) = \begin{cases} \sum_{i=1}^{n} \mathbf{a}_i \phi(||\mathbf{x} - \mathbf{x_i}||) + \mathbf{p}_k(\mathbf{x}) & I_P(\mathbf{x}) = 1 \\ \sum_{j=1}^{m} \mathbf{b}_j \phi(||\mathbf{x} - \mathbf{y_j}||) + \mathbf{q}_k(\mathbf{x}) & I_Q(\mathbf{x}) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $\mathbf{a}_i$ and $\mathbf{b}_j$ are the 2D coefficients, which are found in the same way as described before.

To represent the discontinuity, we obtain a transparency map, A, that is used to modulate the opacity of the voxels when rendering the volumetric object, as described in [CSC06b]:

$$A(\mathbf{x}) = \begin{cases} 1 & I_P(\mathbf{x}) = I_P(\mathbf{x} + D(\mathbf{x})) \wedge I_Q(\mathbf{x}) = I_Q(\mathbf{x} + D(\mathbf{x})) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

An example is shown in Fig. 2, where a cut (shown as a yellow line) divides the grid guide in two regions. When applying a deformation, the grid is split as it crosses the boundary defined by the cut, effectively forming a discontinuous deformation. Fig. 2(d) also shows the result of applying the displacement to the bar dataset. Note also that a complex deformation can be found by interpolating the displacement values down to zero, in order to create a smoother cut (Fig. 2(e)), using the modulation rule for 3D extrusion.

4.3. Complex 3D Deformation

Our approach simplifies the deformation problem by considering it along a 2D direction, rather than in 3D space. This makes it very fast for large datasets. However, it also limits the types of deformation that can be obtained with our approach. There are two ways to obtain complex deformations in 3D space, beyond the simple rules of replication, modulation and interpolation. The first approach is to apply progressively deformation along different 2D planes. After a deformation has been applied, the user may rotate the volume and apply a different deformation along a different plane. Deformations are then combined into a single transformation, via blending. A different approach is to extend Eq. (2) to 3D and allow the user to handle control points in 3D space. Eq. (2) becomes:

$$\mathbf{D_{3D}}(\mathbf{x}) = \sum_{i=1}^{N} \sigma_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \mathbf{q}_k(\mathbf{x}) \quad (9)$$

where $\sigma_i = [\alpha_i, \beta_i, \delta_i]^\top$ is a tuple of coefficients in 3D and $\mathbf{q}_k$ is a polynomial of degree k. In this case, $\mathbf{q} = \mathbf{q}_0 + \mathbf{q}_1 x + \mathbf{q}_2 y + \mathbf{q}_3 z$. This equation can be incorporated directly into a 3D volume rendering system. However, it is computationally more costly than our 2D deformation mechanism. An example is shown in Fig. 6(a,b), where the CT head dataset is morphed directly in 3D. Another example is shown in Fig. 6(c,d), where a horse model, represented volumetrically as an implicit surface, is deformed directly. A fast mechanism for the solution and application of 3D interpolants is currently being sought.

5. Results and Applications

Our approach enables the user to interactively illustrate the type of deformation that is desired. The illustration is in 2D, however, the resulting deformation is 3D. By allowing the user to directly define the deformation, many new types of deformations/cuts/illustrations can be created. One example is the use of deformation in the analytical exploration of morphology and the study of evolution, as pioneered by D'Arcy Thompson in his book "On Growth and From". In his work, he uses the deformation of a 2D Cartesian grid to explain and illustrate the morphological differences between different animal species. In Fig. 3, an example of his illustrations is depicted, where a rectangular grid is used to inscribe an illustration of the *Polyprion* species. Another species, *Antogonia carpos*, can be inscribed in a non-linear grid, obtained by deforming the rectangular grid. This method of deformation is essentially an interpolation mechanism given the displacement of control points, which correspond to matching features of interest. In our case, we have adapted this idea to the deformation of the carp dataset. We obtained first a 2D image using MIP and inscribed it into a 2D grid. After deforming the image based on the deformation grid of the reference illustration, we generated a 3D displacement that smoothly interpolates the 2D displace-

ment along the Z direction. Results are shown in Fig. 3. Unlike a 2D illustration, the deformation of a volumetric object allows the exploration from multiple viewpoints. This approach can be extended to evolutionary morphing simulation directly on volumetric objects, extending the surface morphing approach introduced in [WAA*05].

Fig. 4 shows an example of using this approach to kinematic movement of a CT scan of a knee. A 2D image is obtained by computing an MIP projection. A set of control points are defined along the contour of the distal end of the femur and another control point is defined in the lower part of the calf. We then displace this control point to simulate a bending of the knee. We see that affine transformations are not sufficient for a plausible deformation, whereas constraining the coefficients to be as-rigid-as-possible gives a more natural bending. Note that the lower portion of the tibia and fibula bones are deformed rigidly. The obtained displacement is used to deform the volumetric model. Fig. 5 shows an application of this method for the animation of volumetric objects. In this case, a 3D displacement is obtained at different time steps to simulate a complex deformation. Note that large displacements can be simulated with a small number of control points. Because each time step requires a 3D displacement, this could be expensive. However, since each 3D displacement is obtained from a 2D displacement (using the replication method), they can be stored in a single 3D displacement. In such case, the z dimension corresponds to time instead of a spatial dimension. A number of frames are shown. Fig. 7 shows an example of cutting the abdomen dataset using decoupled RBFs. In this case, the control points are divided into two regions I_P and I_Q, to the left and right of the cut, respectively. By moving the control points in opposite directions, a cut is obtained. To show the effect of cutting in 2D, we use a background image of the abdominal organs. In Fig. 7(d), we perform deformation on the 3D dataset by obtaining the 3D displacement from the 2D manipulation. Fig. 6 shows two examples of using our approach for direct 3D deformation. In this case, the equations refer to 3D displacement directly, instead of 2D. Although this is computationally more expensive, it produces more complex deformations. Fig. 6(a,b) shows the deformation of a CT head. In this case, the deformation is incorporated with a volume rendering system, so that sample points are warped according to the displacement given by solving the RBF equations. A number of control points are shown. The deformation of volumetric objects are not only useful for CT or MR images, but also can be used to deform surface models. In this case, a volumetric representation of the model is obtained, such as a signed distance field, and then deformed with our approach. Fig. 6(c,d) shows the deformation of a horse model represented implicitly in a $256 \times 256 \times 256$ volume. To maintain the legs fixed, we include a rigidity constraint as the distance to the horizontal plane where the horse is standing, using Eq.(6).

5.1. Implementation Details

In our implementation, we use a combination of CPU and GPU to maximize performance. Because we attempt to obtain complex deformation with a small number of control points, finding the coefficients for the RBF interpolant is done in the CPU. We use the open source linear algebra package Lapack to solve the linear systems of equations. In our experiments, we have deformations from less than 8 control points to up to 32, with a solving time of 0.25 to 2.4 milliseconds. We use a Pentium M processor 1.6 GHz with a Quadro FX Go1400 with 256 MB of texture memory. The generation of the displacement map, however, is done on the GPU, by evaluating Eq.(2) on a per-pixel manner. This is considerably faster than doing it on the CPU. For a 256×256 image, generation of the displacement map takes about 20 milliseconds on the GPU, while it goes from 633 (8 control points) up to 2200 milliseconds (32 control points) in the CPU. Our GPU implementation has proved to be essential for real-time interaction. However, current GPU architectures limits us to a maximum of 32 control points. Although complex deformations can be obtained with 32 control points, we believe that this will be improved on new generations of GPU's.

6. Conclusions

We have presented a method for specifying deformations on a volumetric object based on scattered data interpolation. Rather than directly transforming voxels or elements in a proxy mesh, we reduce the problem to the deformation of 2D points. This reduces the complexity inherent with the transformation of complex 3D objects. Having a 2D image as the input for deformation is consistent with many illustration procedures, which often use references images as inspiration for the definition of a complex deformation. In our approach we let the user define a set of control points in the image and displace them in 2D space. Scattered data interpolation, based on compactly supported radial basis functions, is used to create a smooth displacement map. We showed how complex deformations may require additional constraints, such as similarity or rigidity constraints, which results in more plausible deformations, or the decoupling of RBFs to support discontinuities, needed for cuts. We also showed how a 3D displacement map is obtained by extruding the 2D displacement based on simple rules defined by the user. We also showed how this approach extends directly to 3D deformation, but at a considerable computational cost. We believe that this method has a great potential as a new way of interacting with volumetric objects.

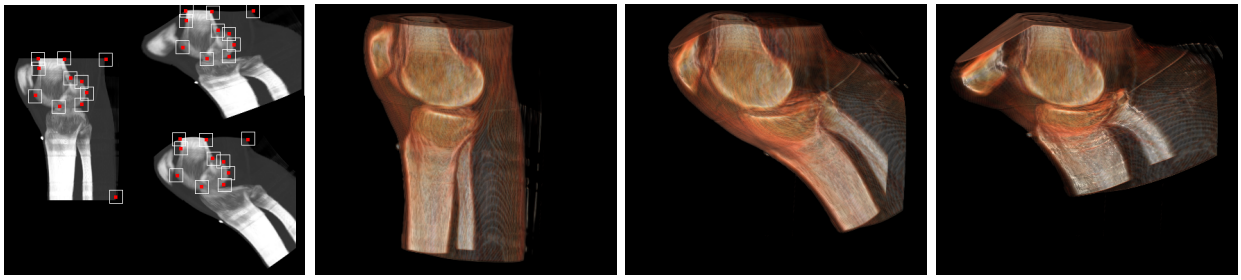
References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *SIGGRAPH: Proc. of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 157–164. 2

[BK05] BOTSCH M., KOBBELT L.: Real-time shape editing us-

(a) (b) (c)

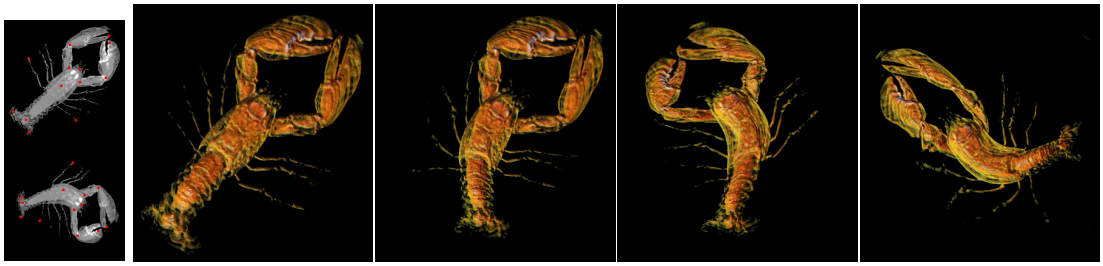
Figure 3: *Application of our approach to Morphology Illustrations. (a) Reference illustrations (From "On Growth and Form" [Tho61]) and 2D images mimicking the illustration (b) Original carp dataset (c) Deformed carp dataset.*



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

Figure 4: *Bending of CT knee dataset. (a) 2D control points and affine and rigid deformations (b) Original volumetric object (c) Deformation with rigidity constraints (d) Deformation without rigidity constraints.*

.



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

Figure 5: *Animation of Lobster dataset. (a) 2D deformation with control points (b-e) Sample time frames of the animation.*

.

ing radial basis functions. *Computer Graphics Forum 24*, 3 (2005), 611–621. 2

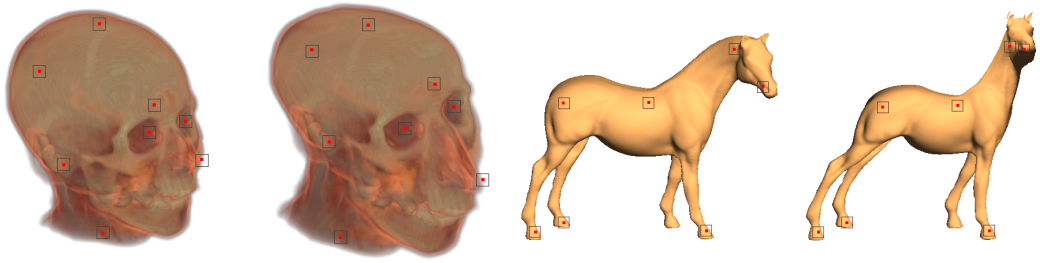
[CSC06a] CORREA C., SILVER D., CHEN M.: Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (September-October 2006), 1069–1076. 1, 2, 4

[CSC06b] CORREA C. D., SILVER D., CHEN M.: Discontinuous displacement mapping for volume graphics. In *Proc. of Volume Graphics* (2006), pp. 9–16. 1, 2, 4, 5

[CSW*03] CHEN M., SILVER D., WINTER A. S., SINGH V., CORNEA N.: Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proc. Volume Graphics* (2003), ACM Press, pp. 35–44. 2

[FHSR96] FANG S., HUANG S., SRINIVASAN R., RAGHAVAN R.: Deformable volume rendering by 3D texture mapping and octree encoding. In *Proc. IEEE Visualization* (1996), pp. 73–ff. 2

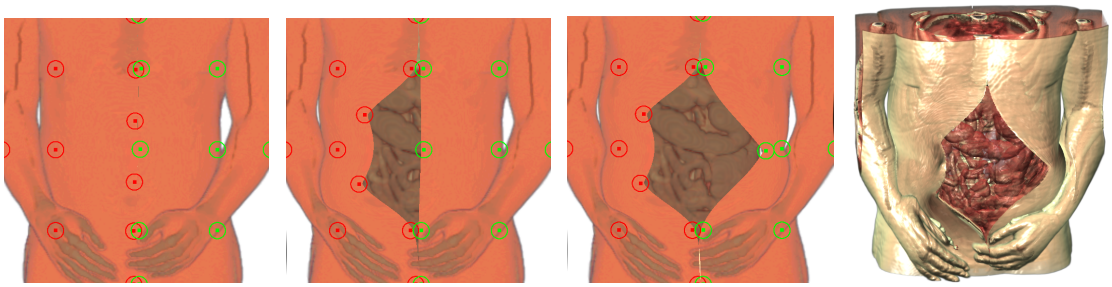
[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J.: As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH* (2005), pp. 1134–1141. 4



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

Figure 6: *Complex deformations in 3D (a)-(b) Deformation of CT Head volume. (c)-(d) Deformation of horse model treated as a volume (implicit surface).*

.



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

Figure 7: *Cut deformation of abdomen dataset using decoupled RBFs. (a)-(c) 2D deformation, where cut divides the control points into two sets, marked as red and green. By moving the points in opposite directions, a cut is achieved. A background image of the abdominal organs is used to show the interior. (d) A 3D deformation after obtaining the displacement from the 2D manipulation.*

.

[KSSH02] KOJEKINE N., SAVCHENKO V., SENIN M., HAGIWARA I.: Real-time 3D deformations by means of compactly supported radial basis functions. In *Eurographics 2002. Short Presentations* (2002). 2

[KY97] KURZION Y., YAGEL R.: Interactive space deformation with hardware-assisted rendering. *IEEE Comput. Graph. Appl. 17*, 5 (1997), 66–77. 2

[LGL95] LERIOS A., GARFINKLE C. D., LEVOY M.: Feature - based volume metamorphosis. In *SIGGRAPH: Proc. of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM Press, pp. 449–456. 2

[LHH97] LITTLE J. A., HILL D. L. G., HAWKES D. J.: Deformations incorporating rigid structures. *Comput. Vis. Image Underst. 66*, 2 (1997), 223–232. 2, 4

[MTB03] MCGUFFIN M. J., TANCAU L., BALAKRISHNAN R.: Using deformations for browsing volumetric data. In *Proc. of IEEE Visualization* (October 2003), pp. 401–408. 1

[RM93] RUPRECHT D., MÜLLER H.: Free form deformation with scattered data interpolation methods. 267–281. 2

[RM95] RUPRECHT D., MÜLLER H.: Image warping with scattered data interpolation. *IEEE Comput. Graph. Appl. 15*, 2 (1995), 37–43. 2, 3

[RSSSG01] REZK-SALAMA C., SCHEUERING M., SOZA G., GREINER G.: Fast volumetric deformation on general purpose hardware. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (New York, NY, USA, 2001), ACM Press, pp. 17–24. 2

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3 (2006), 533–540. 2, 4

[Tho61] THOMPSON D. W.: *On Growth and Form*. Cambridge University Press, 1961. Abridged ed. J.T. Bonner. 7

[WAA*05] WILEY D. F., AMENTA N., ALCANTARA D. A., GHOSH D., KIL Y. J., DELSON E., HARCOURT-SMITH W., JOHN K. S., ROHLF F. J., HAMANN B.: Evolutionary morphing. In *IEEE Visualization* (2005), p. 55. 6

[Wen95] WENDLAND H.: Piecewise polynomial, positive defined and compactly radial functions of minimal degree. *AICM 4* (1995), 389–396. 3

[WRS01] WESTERMANN R., REZK-SALAMA C.: Real-time volume deformations. *Comput. Graph. Forum 20*, 3 (2001). 1, 2