Heterogeneous Volume Deformation and Animation

Authoring with Density-Aware Moving Least Squares

Soonhyeon Kwon   Younguk Kim   Kihyuk Kim   Sungkil Lee

Sungkyunkwan University

Suwon, South Korea

{soon34564,ywkim91,kihyuk,sungkil}@skku.edu

Abstract

This paper presents a novel heterogeneous volume deformation technique and an intuitive volume animation authoring framework. Our volume deformation extends the previous technique based on moving least squares with a density-aware weighting metric for data-driven importance control and efficient upsampling-based volume synthesis. For user interaction, we present an intuitive visual metaphor and interaction schemes to support effective spatiotemporal editing of volume deformation animation. Our framework is implemented fully on graphics processors and thus suitable for quick-and-
easy prototyping of volume deformation with improved controllability.

**Keywords:** volume, deformation, heterogeneous volume, animation, moving least squares
Introduction

Scientific volumetric datasets (hereinafter, *volumes*) often require to be visualized to facilitate their exploration and inspection tasks. The volume datasets are not directly associated with visual appearance, and thus, volume visualization techniques map their scalar or vector fields to optical/physical properties to effectively visualize them. With the great strides made in graphics processing units (GPUs), *direct volume rendering* [1], involving entire voxels in visualization, has drawn growing attention, which can provide instant feedback for interactive volume visualization.

Direct volume rendering bases visualization typically on a regular spatial structure (e.g., 3D grids or textures [2]) to sample a scalar field, and the samples are transformed to physical/optical quantities via a transfer function [3]. An appropriate design of transfer functions is of crucial importance for perceptually effective visualization. A simple mapping to opacity or colors works, but the design of sophisticated transfer functions has been challenging [4]. An alternative approach has focused on relaxing constraints on the regular spatial structure [5], which was partly inspired by *focus+context* (F+C) distinction in information visualization (see [6, 7] for overview). It has shown that spatial modulation (e.g., distortion or deformation) can also be used to emphasize/reveal semantically important features (*focus*) while preserving spatial relationships with neighborhoods (*context*).

Early volume deformation used physics-based approaches such as energy conservation [8] and fluid dynamics [9]. While physically faithful, simulation parameters are generally hard
to tune as desired due to their indirect effects on appearance. For more intuitive control of complex deformation sequences, non-physically-based deformation, based on proxy structures (boxes or points) and their keyframing, has been proposed [10, 11, 12]. In particular, volume deformation based on the scattered data interpolation such as moving least squares (MLS) [13] has proved its effectiveness owing to its ease to control, on which we focus in this work.

This paper presents an effective and versatile framework for volume deformation and animation. Our approach extends the previous MLS approach [13] with a novel density-aware weighting metric and an upsampling scheme for efficient volume synthesis. Voxel density in our approach serves as an importance for data-driven deformation control. Also, we apply our spatial deformation technique to encompass temporal volume animation and its editing on the basis of intuitive visual metaphors and keyframing, which improves our preliminary work [14]. This visual interaction scheme gives a user informative visualization of amounts and direction of deformations, which leads to a more general and convenient authoring system of animated volumetric sequences. Figure 1 shows an overview of our framework. The major contributions of our approach can be summarized as follows:

- a novel weighting solution for density-aware volume deformation;
- an efficient technique for MLS deformation synthesis based on volume upsampling;
- a spatiotemporal visual metaphor and interaction schemes for intuitive volume deformation editing.
Related Work

In this section, we briefly review the previous studies on F+C volume visualization, volume deformation, and direct volume manipulation.

**Focus+Context Volume Visualization**

Common F+C approaches in volume visualization emphasize some parts of volume (usually, details to observe) by spatially distorting them (e.g., bifocal and fisheye displays), while maintaining proximal relationships at a reduced spatial resolution [5]. Excessive distortion can be perceived, and a later technique used a constrained bound to minimize the distortion [15]. Non-deformative techniques have also been proposed, reducing context visibility with opacity changes [16]. In our heterogeneous volume deformation, analogy with F+C visualization can be found in data-driven importance control (similarly to focus definition). The current formulation leads to better visibility with weaker densities, but can be reversed.

**Volume Deformation**

The most common approach for volume deformation is manipulation of proxy geometries instead of directly controlling volumetric fields. Following the seminal work of Sederberg and Parry on solid geometric objects [17], the transformation of bounding boxes or tetrahedra enabled easy but coarse controls [2]. Local controllability was later improved by piecewise linear patches [11], a volume proxy mesh [12], and volumetric lattices [18].
Concerning animations of human or mammals, skeletal animation of volumes can be an effective alternative for deformation. The process involved here is similar to those of skinning of human meshes; the animation is specified with endpoints, followed by inverse kinematics [19]. While controls are efficient, the reconstruction triggered by skeletal movements can cause heavy processing. Since binding voxels to the skeletons might not be effective, typical drawbacks of skinning are also exhibited, including collapsing joints.

Another mainstream of volume deformation is scattered data interpolation, which commonly employs radial basis functions (RBFs). Its free-form deformation is a good basis for smooth and plausible deformation. Early approach used octree-based morphing [10] and the linear interpolation of 2D slices [20]. A notable recent approach is MLS-based deformation, where a limited number of control points allow us to attain similar results to regular grid-like structures. It has served as a basis in many areas to support quick-and-easy free-form editing. It was initially introduced for image editing [21], but later extended to 3D volumes [13]. The basic formulation uses affine transformation, and further constraints (e.g., similarity and rigidity) induced more plausible outcomes in certain cases.

Our work extends the basic MLS-based deformation [13] to encompass diverse scenarios of volume deformation, including heterogeneous deformation dependent upon local density distribution, non-spatial attributes, and volume animation authoring.
Direct Volume Manipulation

Direct (without relying on proxy structures) volume manipulation was also proposed, such as a sculpting interface [22] and “editing by painting” [23]. The sculpting interface also realized temporal interpolation using the regularized Minkowski operators of individual keyframes. However, individual keyframes require to store entire volume datasets, and memory requirement is too costly in practice; our animation framework stores only control points. In the direct volume editing [23], opacity values control the addition or removal of existing voxels. The approach is fast and flexible for many volume editing operations including coloring, carving, enhancement, and annotation. A plausible outcome requires non-trivial efforts and skilled designers, and further, temporal interpolation would be hard to achieve for complex editing.

Background: Moving Least Squares

In this section, we review the previous MLS deformation models for scattered interpolation of image pixels [21] or volume elements [13]. In addition to the previous MLS formulation, we propose its simple extension to non-spatial attributes. This model serves as a basis for our volume editing framework for more general volume editing.
Spatial Moving Least Squares

The original formulation of MLS for image deformation [21] is based on the forward mapping, which derives deformed positions from source control points. In general, the forward mapping might suffer from incomplete holes, unless sufficient source samples are provided. So, polygonal rasterization is typically used for filling the holes.

However, the forward mapping of a 3D volume (typically using GPU rendering) leads to a significant challenge, because the transformed polygons may lie on 3D planes crossing multiple slices in a 3D texture; such an approach does not fit well to the current GPU pipeline. For this reason, the previous approach [13] used backward mapping, and also, we take the same approach.

In MLS interpolation, the spatial deformation of scalar fields are controlled by a set of control points \( \{x\} \) and their target points \( \{y\} \). While \( \{x\} \) are usually fixed, \( \{y\} \) are altered with user interaction—in our framework, \( \{x\} \) can also move, which allows us to define a minimal set of control points. Let \( u \) and \( v \) be an arbitrary position and its deformation, in which interpolation will be applied. Unlike the typical least squares, an MLS interpolation at a point \( v \) works with its local weights \( \{w_i\} \); hence, it is called the “moving” least squares and considered a special case of the classic weighted least squares. Typically, a local weight \( w_i \) is based on radial distance functions. For instance, a distance metric can be \( w_i = 1/d_i \), where

\[
d_i = |v - y_i|^n.
\]

\( n \) is the power of a square-rooted distance; a typical choice of \( n \) is 2–4. Note that the weight
is computed on the deformed positions, since we follow the backward mapping.

Based on the weight function, weighted centroids of \( \{x\} \) and \( \{y\} \) can be defined as

\[
\hat{x} = \frac{\sum_i w_i x_i}{\sum_i w_i} \quad \text{and} \quad \hat{y} = \frac{\sum_i w_i y_i}{\sum_i w_i}.
\]

Assuming \( \{x\} \) and \( \{y\} \) are associated by affine transformations, the problem of MLS becomes the minimization of the following objective function:

\[
\sum_i w_i |p_i - M q_i|^2, \tag{2}
\]

where \( p_i = x_i - \hat{x} \) and \( q_i = y_i - \hat{y} \), and \( M \) is a matrix linearly transforming deformed points to the source points.

The closed-form solution of \( M \) that minimizes Eq. (2) can be derived with the classic normal-equation solution as

\[
M = \left( \sum_i q_i^T w_i q_i \right)^{-1} \sum_j w_j q_j^T p_j. \tag{3}
\]

Based on this \( M \), the spatial interpolation \( f_s \) at a point \( v \) can be written as

\[
f_s(v) = \sum_j T_j(v) p_j + \hat{x}, \tag{4}
\]

where \( T_j(v) \) is a scalar defined to be

\[
T_j(v) = (v - \hat{y}) \left( \sum_i q_i^T w_i q_i \right)^{-1} w_j q_j^T. \tag{5}
\]

We refer the reader to [21, 13] for more details of derivation and mathematical properties.

The derivation above implies interpolation with general affine transformations, which includes non-uniform scaling and shear in addition to rotation and translation. Constraints
such as similarity and rigid transformation have shown to often better preserve appearance for real-world objects [21, 13]. In our framework, non-spatial attributes and unrealistic cases are also manipulated, and thus, we will base our further extensions on the general affine transformation.

**Non-Spatial Moving Least Squares**

A volumetric representation is typically an input to raster display systems, while images are usually outcomes of (real or virtual) imagery procedures. Hence, when manipulating volumes with MLS, there are more freedoms and potentials over typical spatial deformation. Nonetheless, the previous models have been focused mostly on spatial deformation and its constraints, and thus, it was hard to directly apply them to general volume animation editing.

We here additionally present our straightforward extension to the spatial moving least squares. Whereas Eq. (4) concerns only spatial deformation, it can be applied to the general attributes including non-spatial attributes (e.g., intensity, color, and density). Let general attributes to be interpolated be \( \{z\} \). Similarly to the spatial centroids, the centroids of the general attributes can be

\[
\hat{z} = \frac{\sum_i w_i z_i}{\sum_i w_i},
\]

which derives displacements \( r_i = z_i - \hat{z} \) with respect to \( z_i \).

When the attributes are interpolated based on the spatial deformation, their general MLS
interpolation $f$ of a point $v$ can be simply defined as:

$$f(v) = \sum_{j} T_j(v)r_j + \hat{z}. \quad (7)$$

**Moving Least Squares for Heterogeneous Volumes**

This section presents our novel extensions of the previous MLS model for heterogeneous volume deformation. Previously, the incorporation of heterogeneity (from actual materials) in MLS deformation has hardly been explored; in the absence of deformation, selection of multiple transfer function can be another alternative [24]. Our extensions to MLS model include density-aware distance weighting. This leads us from the simple editing of a volume dataset to flexible volume animation with fine control, further with the temporal control of volume animation editing.

We note that our approach is data-dependent and reflects the density distribution of volume datasets to deformation. This makes a significant distinction from the previous (control-dependent) approaches (e.g., as-rigid-as-possible MLS deformation [25, 18]).

**Density-Aware Distance Weighting**

It is often desired for heterogeneous volumes to suppress propagation of deformation across volume segments of different materials. For instance, where a volume consists of a skull and skin tissues, we may want to apply strong deformation to the skin but little to the skull. However, deformation schemes based on control points usually do not involve the dataset
characteristics in deformation.

To support such heterogeneous deformation, conventional models (for homogeneous deformation) might require explicit hints such as segmentation of volumes. This, however, poses additional challenges in identification of anatomical/structural boundaries of inhomogeneous sub-volumes. Though usually considered to be data-dependent, structural differences in volume data are pronounced in those of density (or intensity).

We fulfill the goal of applying different deformations to inhomogeneous structures by relying on the density or intensity of volumes yet without an explicit segmentation process. Our key idea is to define the distance to be density-dependent. Thereby, structures nearby dense regions become less sensitive to the deformation, while structures with weak densities are deformed more; depending on the application, this can be defined differently.

In the framework of MLS deformation, we realize the idea by scaling $d_i$ in Eq. (1) with the density accumulation from $v$ to $y_i$. More precisely, we make denser structures have higher weights, equivalent to be shorter in distance. Thereby, we redefine the density-aware distance metric as

$$\frac{1}{\rho(v, y_i)} |v - y_i|^n,$$  \hfill (8)

where $\rho(v, y_i)$ is a density accumulation from $v$ to $y_i$.

The density accumulation $\rho(v, y_i)$ is generally hard to analytically measure. We thus rely on a numerical approach using ray casting defined as

$$\rho(v, y_i) = \frac{1}{\phi(v, y_i)} \int_v^{y_i} I(t)dt,$$  \hfill (9)
where \( I(t) \) is the density or intensity at a particular position \( t \) along a ray. \( \phi(v, y_i) \) is a normalization factor, which makes homogeneous regions behave similarly. As for the choice of \( \phi(v, y_i) \), we typically use the minimum density value of the ray segment \([v, y_i]\), making dense rays have shorter distances in heterogeneous regions. In other words, dense structures around heterogeneous boundaries have stronger effects in interpolation, leading to less deformation of their neighbors. In addition, we note that this accumulation is different from the common formulation of ray casting [1]. We use a view-independent integration, and thus, do not have to deal with ray extinction.

A problem here is that we cannot know the internal densities from \( v \) to \( y_i \) in advance of obtaining the deformation—this occurs because we rely on the backward mapping. In other words, we need to obtain the density accumulation in the deformed output volume, but this is not available when evaluating deformation. We solve this problem using numerical iteration.

**Iterative-Search Solution for Backward Mapping**

In order to find \( \rho(v, y_i) \), we may repeat volume synthesis to cast rays for deformed volumes (which cannot be known before deformation). However, this would be very costly, because we need multi-pass volume synthesis requiring to create new volumes for each pass.

The best analogy of \( \rho(v, y_i) \) can be found from \( \rho(u, x_i) \). Since the affine deformation preserves linearity, \( \rho(u, x_i) \) scales well with \( \rho(v, y_i) \). More importantly, \( \rho(u, x_i) \) uses the source volume for ray casting. Hence, this enable a single-pass volume synthesis (for each
slice), bringing significant performance gain. As a consequence, we use the distance metric

\[
\hat{d}_i = \frac{1}{\rho(u, x_i)} |v - y_i|^n, \tag{10}
\]

which yields the same effect as \(\rho(v, y_i)\) is used. Note that \(|v - y_i|\) can still be easily evaluated.

However, we cannot still know \(u\) in advance to the deformation, but it can be iteratively found with a numerical search for \(u\). For this, we use the fixed-point iteration, as an extension to the 2D problem [26]. Given a voxel \(v\), its displacement to the source position \(u\) can be

\[
\delta(u) = v - u = v - f_s(v). \tag{11}
\]

Since a true solution to \(u\) is unknown, we iterate with its estimate \(u_k := f^k_s(v)\), where \(f^k_s(v)\) is a backward mapping weighted with \(\rho(u_k, x_i)\). Then, updating in the fixed-point iteration can be

\[
u_{k+1} \leftarrow w(u_k) := v - \delta(u_k), \tag{12}
\]

where \(w\) is a new estimate and \(u_0\) is a seed for iteration. A fixed point at convergence makes \(\hat{u} = w(\hat{u})\), leading to \(v = \hat{u} + \delta(\hat{u})\); that is, \(v\) is moved from \(\hat{u}\). As a result, we use \(\hat{u}\) convergent to the true solution \(u\).

To facilitate efficient search, we initialize \(u_0\) using the moving least squares without density-aware weighting. In our experiments, 2–3 iterations are practically sufficient for moderate convergence. As for implementation, this iteration can run in a single shader pass for each slice, which avoids redundant slice processing.

In case there is a discontinuity in scalar fields, the iteration may not converge [26]. The
problem can be avoided by segmenting the volume around discontinuity and doing per-segment search, similarly as done in [26]. In practice, our experiment did not encounter such a problem owing to smooth spatial variation from MLS interpolation.

**Volumetric Density Weighting**

Unlike when ray casting is used for high-fidelity rendering, coarser yet fast ray casting would suffice this interpolation purpose. However, in case where there is a very thin structure of high density in a volume, it may not be encountered in the accumulation. To cope with such cases (i.e., not to miss thin dense geometries), it is necessary to narrow down the sampling interval during the ray tracing, but this is very costly.

Our idea is to broaden the coverage of voxel accumulation but with still coarser interval, similarly to the voxel cone tracing [27]; see Figure 2. This conic ray casting scheme uses coarser samples pre-integrating fine-grained samples of high densities, and thereby, greatly reduces cost for ray casting. This scheme can be easily implemented by building 3D mipmap textures (for pre-integration) and controlling the level of details in texture lookup.

**Efficient Volume Deformation Synthesis**

The model of MLS deformation is weighted locally, and requires to evaluate the weighting metric and solve least squares for every voxel. This leads to significant amounts of processing, deteriorating real-time interactivity. In this section, we propose an efficient scheme for volume
synthesis based on the distinction of a detail volume and sparse control volume, leading to efficient multi-pass reconstruction of deformation.

A similar idea was previously discussed in part in the context of the piecewise linear patches [11] (using the texture values as texture coordinates in a different resolution). However, their idea was not realized due to the limited functions of early GPUs. We here revisit and improve the idea in terms of MLS deformation by separating controls and details in volume deformation. We also analyze the performance benefit of this scheme in results. As a preview of the results, we allude that this scheme does not necessarily help all the cases; only when MLS interpolation is costly (e.g., our heterogeneous deformation), this way improves performance.

**Downsampling with Control Volumes**

The MLS deformation usually defines sparse control points in comparison to the number of voxels. The sparsity leads deformation fields to smoothly vary in space. Hence, approximating the deformation in lower spatial frequency does not make a big difference in resulting outcomes.

Based on the observation, we propose an efficient scheme for volume deformation that defines *detail volume* (i.e., a source volume) and *control volume* (i.e., deformation fields as a set of texture coordinates stored in a volume) in different resolutions. In other words, we store the deformation into a lower resolution of volume, and apply upsampling when
synthesizing the output volume. This significantly reduces the amounts of the least-square optimization. In our case that uses ray casting in the distance metric, the difference becomes more apparent.

Figure 3 shows an example of such a splitting scheme. While the control volume was synthesized in a lower resolution, the resulting volume output becomes similar to that synthesized in a higher resolution. Typically, peak signal-to-noise ratios (PSNRs) reside within marginally perceptible ranges (roughly 26–29 dB).

**Volume Upsampling**

The control volume synthesized with MLS has a lower resolution than the detail volume (i.e., a source volume), and thus, it needs to be upsampled to yield the final output volume. The down-sampling factor is relatively low (e.g., a half for each dimension), and a usual Gaussian upsampling would be good for this purpose.

However, a simple upsampling of the control volume would sacrifice the resulting quality too much for performance improvements. Care has to be taken around sharp edges of the detail volume, which may require a sophisticated edge-preserving upsampling such as joint bilateral upsampling [28].

A better solution we propose is to rely on the upsampling of spatial positions (i.e., 3D texture coordinates of the source volume to be mapped) instead of coarser intensity and attributes. This idea is analogous with the differences between Gouraud shading and Phong
shading in typical polygonal rendering. In other words, we fetch the source attributes after upsampling of source positions rather than directly upsampling the intensities and attributes.

For implementation, we used a typical 3D Gaussian upsampling. Even separable filtering further accelerated the upsampling, since the Gaussian kernel is isotropic.

**Spatiotemporal Volume Editing**

In this section, we describe our volume editing framework and visual interaction metaphor to support editing of spatiotemporal deformation as well as non-spatial attributes.

**Control Point Definition**

As already alluded to, targets to be edited in our framework include spatial attributes (local position and source/target transformations), non-spatial attributes (lightness, hue, and density), and temporal duration for each key frame. Positions and displacements of control points are defined by two matrices that transform the source and destination positions. That is, our framework even enables to move the source points, which is very helpful in reducing the number of control points for complex editing. As for the non-spatial attributes, we used lightness, color, and density, but other attributes can also be seamlessly integrated for extension.

When a new keyframe is created, control points are replicated with its own duration. Unlike some of the previous work [22], our framework does not duplicate the entire volume
Visual Metaphor

We visualize the amounts and directions of deformation using an intuitive visual metaphor. Figure 4 shows an example. Source positions and destination positions are colored gray and green, and connected with gray line, respectively. When a control point is selected, the line is highlighted with green color wiring the source and destination positions. Similarly to other free-form editing tools (common in images), our interaction metaphor is particularly effective in rapid prototyping of spatial deformation.

In our implementation, we use a fixed number of control points for different keyframes (e.g., 16 control points). The points are initialized as aligned with the bounding volume with inner offsets. Since our tool allows us to move even the source control points, a small number of points are usually enough.

Unlike spatial deformation, non-spatial (usually visual) attributes do not require visual metaphor, since the appearance of rendering itself inherently reflects the attributes.

Interaction Scheme

When moving control points, we could typically move them with limiting the translation axis as done in many 3D modeling tools. This is good for precise translation, but often not much intuitive without visualizing the Cartesian axes at the same time.
We instead employ a more intuitive interaction scheme, which is similar to panning of a camera common in 3D navigation. In general, movement along the viewing vector is hard to perceive, since its screen movement is subtle. Hence, we constrain the movements of control points (typically using a mouse) in a plane parallel to the viewport plane; see Figure 5 for its illustration. By this way, the user can efficiently move the control points as closely as one sees on the screen.

Such an interaction scheme can also be easily mapped to the editing of non-spatial attributes. For instance, the change of lightness is mapped to the vertical movement on the screen. Likewise, hue values are mapped to both the horizontal and vertical movements. In our case, a left-to-right movement is reflected to green-to-red, and a bottom-to-top movement blue-to-yellow; we use CIE L*a*b* color space (see Figure 6 for an example).

**Spatiotemporal Interaction Metaphor**

We lastly present the spatiotemporal editing for keyframe animation, which integrates all the previous interaction schemes with spatiotemporal visual metaphor. In order to facilitate authoring of (temporal) volume animation, we employ keyframes and their durations (or time intervals).

While a typical keyframe editing framework visualizes shapes at a particular time stamp, our framework visualizes the preceding and succeeding positions at the same time (see Figure 7). In the figure, positions of three consecutive frames are visualized simultaneously.
using Y-shaped connectors. This is very useful for intuitively understanding temporal evolution of spatial deformation. Although it is possible to visualize all the control points with this visualization, it is better to focus on a selected control point to reduce visual clutter.

Unlike the visualization of spatiotemporal deformation, it is not straightforward to visualize the temporal transition of non-spatial attributes. One might overlay lightness or colors along different frames, but our efforts did not reach any good solutions due to too much complexity involved in the visualization.

**More Examples and Performance Analysis**

In this section, we demonstrate the utility of our framework as a versatile mediator subserving free-form editing of homogeneous and heterogeneous volume deformations and their animation. We also report performance evaluations attained by our different speed-up solutions.

**More Examples**

Figure 8 shows animated sequences of five volume datasets (the *Skull, Abdomen, Foot, Smoke*, and *Lobster*); also, see the accompanying video clip. The *Skull* demonstrates a typical homogeneous deformation and its animation by applying a usual MLS deformation with inverse-distance weighting. The *Abdomen* shows another example of complex homogeneous deformation with complex definitions of control points. The *Foot* shows an example of how
to apply heterogeneous deformation; while bones stay roughly at the same locations, the flesh deforms heavily. The Smoke shows potentials of diverse non-spatial attributes, including lightness, density, and hues. Lastly, the Lobster shows an example integrating spatial and non-spatial deformation together. We note that we did not apply shading to better observe internal structure changes resulting from deformation.

Figure 9 shows a detailed comparison of homogeneous and heterogeneous deformation given the same control points. To facilitate the heterogeneous deformation, control points are located around the dense structures (in the figure, bones). While the homogeneous deformation here behaves similarly to a usual MLS deformation, the heterogeneous deformation strongly suppresses the propagation of deformation fields, better preserving the dense structures. By contrast, the regions far from the bones are still deformed significantly.

As shown in the examples, our framework allows us to effectively author volume deformation and animation for many different scenarios, including spatial, non-spatial, and heterogeneous deformations. Such animation generally requires significant efforts for its editing, and even hardly achievable with a common editing scheme. By contrast, our framework allows volume artists and designers to save time and efforts greatly; in our cases, when trained moderately, it took for a 3D model/volume designer 10–30 minutes for a single shot and 25–80 minutes for the entire animation, which can be thought fairly short, considering the complexity involved in animation. Furthermore, additional attributes can be easily added for extension, since our framework also enables editing of even non-spatial attributes.
Performance Analysis

We implemented our framework with OpenGL API fully on GPU for both the volume rendering and authoring. We assessed the performance of our implementation on an Intel i7 3.0 GHz with an NVIDIA GeForce GTX 980 Ti. The volume rendering for visualization used typical stepwise ray casting (the average step size of 1024). Input and output volume datasets are represented as 3D textures, and editing of output volumes are updated by slice-wise multi-pass rendering (the synthesis of an individual slice took a single rendering pass). When control volumes are used, we used a half resolution for each axis.

The dimensions and appearances of the volume datasets are summarized in Figure 8. The volume datasets of moderate resolution were used for performance evaluation, which ranges from $100^2 \times 40$ to $512^2 \times 174$.

Figure 10 summarizes the results of our experiments, measured in terms of frame time (ms). Regular homogeneous animation (without density-aware weighting of the MLS), the frame time reaches 3–33 ms, which efficiently runs in real-time rates. The use of control volumes, however, deteriorates the performance down to roughly 50 %. The upsampling involved with the control volumes incurs too many texture lookups (e.g., the separable filtering of $5 \times 5 \times 5$ Gaussian kernels requires 15 texture lookups), and hence, the sole use of MLS becomes faster unlike usual expectation.

In contrast, the heterogeneous deformation brought about significant performance degradation, because it requires ray casting for each control point during the animation. Without
our advanced speed-up techniques, the cost increases 2–10× than those of the homogeneous deformation (without density-aware weighting). As for the speed-up techniques, the use of conic ray casting significantly reduces the number of ray leaps for the density weighting, leading to speed-ups of up to 30%. Importantly, the use of control volume is now significantly beneficial unlike those of the homogeneous deformation; the speed-up reaches up to 350% against those without the control volume. Combining the conic tracing and control volume together, the frame time is significantly reduced to interactive frame rates, but still slower than the homogeneous deformation (roughly 50%).

In addition, we report the effect of number of control points, which is one of the major performance factors. The number of control points used 0, 8, 16, 24, and 32 for the test. The heterogeneous animation was used with both the conic ray casting and control volume. Table 1 shows the evolution of performance along the number of control points. It is observed that our algorithm scales very well with the number of control points; however, there is a base overhead for applying deformation in separate passes. Nonetheless, we emphasize that our framework requires a small number of control points, because our framework enables to move even the source control points. In our tests, sixteen points are usually enough for rather complex heterogeneous deformation, as shown in Figure 8.
Discussion and Limitations

One of the common limitations of deformation-based editing tools, including ours, is the lack of volume space expansion. For a relatively large deformation, the framework need to allow expansion of the volume dimension without redundancy. At present, our framework does not support such a feature, since it is not easy to determine how much space requires to be expanded.

The density-aware weighting scheme offers natural convenience of heterogeneous deformation, but significantly degrades the performance without sophisticated techniques. Our present efforts for speed-up proved their benefit, but are still costly. A more efficient weighting solution without relying on ray casting can be a good direction for future work.

Unlike the typical image deformation, the volume deformation needs to be backward-mapped. The forward mapping has more benefit and freedom in terms of clear formulation and weighting schemes beyond our density-aware weighting scheme. However, to our knowledge, any of the previous studies did not find a good solution for such forward mapping based on polygonal rasterization. Finding a viable solution for forward mapping can be also a good direction for further research.

In our framework, the control points are initially aligned regardless of the content of volume data, which is often inconvenient in terms of usability of editing tools. Automatic initialization of control points can be very useful for rapid prototyping. Obviously, reflection of density distributions and other salient attributes can be good starting points.
Deformation based on the MLS inherently yields spatially smooth interpolation. Though, we might need to improve sharpness around specific segments. The common formulation of the volume deformation does not allow us to define sharp features. A more sophisticated deformation scheme would be also useful for good volume animation editing.

Since our framework bases the editing of volume datasets on control points, it is not straightforward to add and define details to the internal sub-volumes within the control points, because atomic control is defined for each control point. A hybrid approach integrating the MLS and direct manipulation [23] would add much flexibility in volume editing.

**Conclusions**

In this article, we presented a novel heterogeneous volume deformation technique and a versatile framework for volume animation authoring. Heterogeneous volume deformation with 3D MLS was realized with a novel density-aware weighting scheme in a data-driven way, followed by speed-up solutions with coarser control volumes. We demonstrated the utility of our framework as a viable prototyping solution for real-time volume editing and animation. Since there is still a room for further improvements in terms of performance, flexibility, and usability, further research is encouraged onto those directions.
Acknowledgments

This work was supported in part by the Mid-career and Global Frontier (on Human-centered Interaction for Coexistence) R&D programs through the NRF grants (Nos. 2015R1A2A2A01003783, 2012M3A6A3055695), the ITRC support program (IITP-2017-2016-0-00312) supervised by the IITP, funded by the Korea Government (MSIP), and Faculty Research Fund, Sungkyunkwan University, 2011. The volume data were provided by the courtesy of Computer Graphics Group, University of Erlangen (the Skull volume dataset), GRaphisch-Interaktive Systeme (GRIS), Wilhelm Schickard Institut für Informatik, Universität Tübingen (the Abdomen, Foot, and Lobster volume datasets), and Physically Based Ray Tracing (the Smoke volume dataset). Correspondence concerning this article can be addressed to Sungkil Lee.

References


Figure 1: Example overview of our framework for volume deformation and its authoring interface: (a) initial control points (green dots) aligned with XYZ axes; (b) homogeneous spatial deformation defined with displacements (gray dots); (c) heterogeneous deformation where bones deform little but fleshes deform more; (d) an outcome with additional changes of non-spatial attributes (e.g., hue and brightness).
Figure 2: Example comparison of density accumulations via (a) linear and (b) conic ray casting. The thin detail of high density is captured with the conic ray casting but not with the linear accumulation.
Figure 3: Comparison of volumes synthesized (a) without and (b) with sparse control volumes for speed-up. (c) shows their absolute pixel-wise differences.
Figure 4: Visual interaction metaphor for spatial volume deformation authoring.
Figure 5: Planar interaction scheme along the translation plane.
Figure 6: Example of alteration of hue values and its interaction scheme.
Figure 7: Temporal evolution of the spatiotemporal visual metaphor along five keyframes. Red, green, and blue dots indicate previous, current, and next key frames, respectively.
Figure 8: Example renderings using our volume deformation animation authoring framework, including homogeneous and heterogenous deformations with non-spatial attributes.
Figure 9: Homogeneous (b) and heterogeneous (c) deformation synthesized with the same control points, given the input volume (a).
Figure 10: Performance measurements evaluated in terms of frame time (ms) for combinations of our different speed-up solutions.
Table 1: Performance evaluation for different numbers of control points, measured in terms of frame time (ms). Measurements without control-point definition indicate rendering-only performance (without deformation processes) as baselines.

<table>
<thead>
<tr>
<th>Control points</th>
<th>Skull</th>
<th>Abdomen</th>
<th>Foot</th>
<th>Smoke</th>
<th>Lobster</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$256^2 \times 106$</td>
<td>$512^2 \times 174$</td>
<td>$256^3$</td>
<td>$100^2 \times 40$</td>
<td>$301 \times 324 \times 56$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>58</td>
<td>34</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>83</td>
<td>50</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>24</td>
<td>26</td>
<td>105</td>
<td>58</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>136</td>
<td>78</td>
<td>8</td>
<td>31</td>
</tr>
</tbody>
</table>