DOI: 10.1111/cgf.12166

COMPUTER GRAPHICS *forum*
Volume 32 (2013), number 8 pp. 122–133

Four-Dimensional Geometry Lens: A Novel Volumetric Magnification Approach

Bo Li, Xin Zhao and Hong Qin

Computer Science Department, State University of New York at Stony Brook (Stony Brook University), Stony Brook, NY, USA
{bli, xinzhao, qin}@cs.stonybrook.edu

Abstract

We present a novel methodology that utilizes four-dimensional (4D) space deformation to simulate a magnification lens on versatile volume datasets and textured solid models. Compared with other magnification methods (e.g. geometric optics, mesh editing), 4D differential geometry theory and its practices are much more flexible and powerful for preserving shape features (i.e. minimizing angle distortion), and easier to adapt to versatile solid models. The primary advantage of 4D space lies at the following fact: we can now easily magnify the volume of regions of interest (ROIs) from the additional dimension, while keeping the rest region unchanged. To achieve this primary goal, we first embed a 3D volumetric input into 4D space and magnify ROIs in the fourth dimension. Then we flatten the 4D shape back into 3D space to accommodate other typical applications in the real 3D world. In order to enforce distortion minimization, in both steps we devise the high-dimensional geometry techniques based on rigorous 4D geometry theory for 3D/4D mapping back and forth to amend the distortion. Our system can preserve not only focus region, but also context region and global shape. We demonstrate the effectiveness, robustness and efficacy of our framework with a variety of models ranging from tetrahedral meshes to volume datasets.

Keywords: focus+context volume visualization, 4D geometry lens, 4D parameterization and mesh editing

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism Visualization

1. Introduction

The rapid advances in three-dimensional (3D) scanning, acquisition and modelling techniques have given rise to the explosive increase of volumetric digital models with extra density information like magnetic resonance imaging (MRI), textured solid models [KFCO*07], [TOII08] or computer aided detection (CAD) models containing materials. The great progresses in graphics processing unit (GPU) rendering, and internet bandwidth, push forward a stronger-than-ever need for visualizing large-scale volume datasets in various science/engineering applications. Meanwhile, the explosive emergence of various types of portable mobile devices (e.g. smart phone) pursues the visualization technique to display large-scale models on a physically limited device screen. It requires us to non-homogeneously rescale different regions while keeping the global shape of models within the screen space.

The traditional method is through the use of 2D screen region-of-interest (ROI) magnification techniques, which functions as 'lens' and offers a good strategy to magnify a local region only. However, compared with magnification on the image projected on the screen, it is more preferable to locally magnify the 3D volume datasets directly. For example, the user can translate, rotate, cut and visualize the dataset from different angles without computing magnification again and again. Magnifying datasets directly is also necessary for many virtual reality applications (e.g. cultural heritage and walkthrough).

From practitioners' perspective, an attractive magnification should address the following quality-centric aspects: *Shape-preserving*—Shape (such as angle, rigidity) plays a crucial role during magnification when improving the visual cognition. The improper magnification distortion may cause serious cognitive confusion. We should preserve the shape of focus region, surrounding context region and global shape simultaneously. *Smooth transition*—Any visual gain from unifying the local detail with the surrounding context may easily be lost if the transition between the focus and context regions is difficult to understand. *Simple*

interaction—In most practical applications, the user only prefers to use simple user sketch (e.g. draw a circle) to enclose the focus region. Our system should thus support such simple interaction.

However, it is a tremendous challenge to optimize the output simultaneously with respect to all of the aforementioned aspects. The most challenging side effect is that: in a 3D world, a local region's magnification inevitably compresses the rest region and leads to distortion. More severely, the conventional methods are more likely to spread the distortion throughout the 3D space. Any optimization technique only moderates but never eliminates distortion. Meanwhile, the existing techniques consider neither shape-preserving nor smooth transition from the rigorous geometry's point of view, thus lens distortions are intolerable when features become sufficiently intricate.

To tackle the above-mentioned challenges, we are inspired by the following idea: Rather than magnifying ROIs and shrinking the rest region in the 3D world, we could increase ROIs' volume in the additional dimension without changing the rest region. Also, it is a well-known knowledge that the differential geometry theory and its practical techniques (e.g. surface parameterization) can handle angle distortion rigorously and quantitatively. In this way, we examine this conventional magnification task from a completely innovative perspective of 3D/4D geometry processing.

To achieve this goal, we propose a framework to simulate 4D lens in order to achieve local magnification while minimizing global angle distortion. Our framework starts from transforming the 3D input into a 4D mesh with an initial fourth dimension for every vertex. Then we conduct 4D deformation which enlarges ROI's volume while keeping the rest unchanged. To visualize this magnified mesh, we automatically deform the mesh back into 3D space for applications. Both steps require us to seek distortion minimization for each individual mesh element during deformation. Specifically, our contributions in this work include:

(1) A framework to address the 3D volume dataset magnification. In contrast to other possible deformation solutions, our method lets the additional dimension's space absorb the volume magnification rather than spreading throughout the nearby space in the original dimensions. Therefore, our result can resemble the original interior texture and the resulting transition between ROIs and the rest is also smooth and seamless.
(2) Techniques for distortion minimization with high dimensions. To achieve this, we propose a piece-wise method to solve the harmonic function on nD tetrahedral mesh. Meanwhile, we develop a flattening method to model the 4D shape flattening back into 3D and preserve the shape.

Our system has the unique feature that we can preserve the shape around both focus region and context region/global shape. Our geometry-based method can also quantify and minimize distortion. Therefore our system can effectively magnify and visualize volume datasets while keeping distortion unnoticeable. Also, we first introduce 4D geometry theory into computer graphics as a practical application and powerful tool for real-world (3D) visualization and modelling.

After discussing related literatures, a framework overview is given in Section 3. On a global view, modelling the 4D magnification in Section 4 is the first stage in our framework, followed by flattening techniques in Section 5. In Section 6, we demonstrate our experimental results and document more comprehensive discussion, respectively.

2. Previous Work

Lens design problem originates from 2D image viewing study. Unlike image resizing [WTSL08], focus + context application requires non-uniform magnification. Optical effect, such as fisheye lens [Fur86], offered an effective navigation and browsing device for various applications [NBM*06]. More sophisticated lenses, like Bier *et al*. [BSP*93], utilized a toolglass and magic lenses to enhance the focal interest features and compress the less interesting regions. Carpendale *et al*. [CCF97] proposed several view-dependent distortion patterns to visualize the internal ROI, where more space is assigned to highlight focal regions. LaMar *et al*. [LHJ01] presented a fast and intuitive magnification lens with a tessellated border region by estimating linear compression according to the radius of lenses and texture information. Recently, Pietriga *et al*. [PA08], [PBA10] provided in-place magnification without requiring the user to zoom into the representation and consequently lose context. Similar to our scheme, Zhao *et al*. [ZZG*12] proposed the geometry method that magnifies the 2D image in 3D space for visualization of 2D image. Unlike the above-mentioned 2D techniques, our ambitious goal is to design lens for 3D volume. To our knowledge, no existing method has attempted to provide the solution yet.

Another interesting prototype in lens design is that deformation and mesh-editing–based methods are recently used for the complicated 3D datasets, including volume data [CS07] [WWLM11] and mesh models [WLT08]. Wang *et al*. [WLT08] presented a method using an energy optimization model for large surface models. Later, they further extended this framework into 3D volumetric datasets [WWLM11]. Zhao *et al*. [ZLWK12] proposed a geometry-based deformation mechanism using moving least-square method for volume data visualization. Inspired by these methods, we utilize the relative geometric method, which originally applies to visualization of 2D datasets like [ZZG*12], targeting to eliminate the local angle distortion and keep the visual continuity for volume datasets.

High-dimension modelling and processing. Unlike other lens design methods, our 4D lens framework utilizes vital geometric modelling techniques like texture transfer and flattening. Texture transfer is in essence a parameterization problem which seeks a one-to-one continuous map between an n-manifold and a target $n-1$ domain with low distortions. Surface parameterization (2D image to 3D triangle mesh) is widely studied because of its importance in graphics and shape modelling. We refer readers to comprehensive survey reports of [FH05], [SPR06] and [HLS07] for various surface parameterization techniques. A main challenge here lies at generalizing 3D techniques to higher dimension. More closely related to our work, volumetric parameterization (3D-to-3D) has gained great interest in recent years and a few related techniques have been developed towards remeshing [LGW*07] or spline construction [MCK].

One key technique in our framework is volumetric mapping. Among all volumetric parameterization techniques, we have high

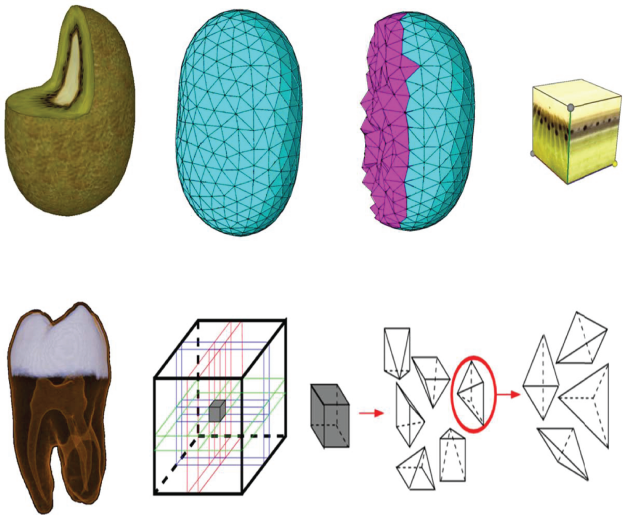


Figure 1: *Inputs of our framework. Top: A 3D solid textured model is a tetrahedral mesh mapped by the colour texture. Bottom: For a volumetric dataset, we partition the space into grids and each grid is uniformly subdivided into tetrahedra.*

interest in using local coordinates to minimize harmonic energy, because it is particularly suitable for parameterization on discrete tetrahedral mesh. The coordinates on the triangle mesh, like [Wac75], [MDSB02], [Flo03], [LLCO08], are fully developed in the conventional surface modelling. As for coordinates on the tetrahedral mesh, Wang *et al*. [WGC*04] proposed a variational coordinate method to parameterize solid shapes over a solid sphere. [LQ12] managed to integrate volumetric local coordinates and interpolation directly into continuous shape representations. Ju *et al*. [JSW05] generalized the mean value coordinates [Flo03] from surface to volume for a smooth volumetric interpolation. One obstacle within these methods, however, is that building the coordinates requires complex geometric algorithms, and these discrete methods always do not converge for complex shape. [LLWQ13] developed simplified local volumetric coordinates based on domain decomposition and regular poly-cube domain construction to simplify the mapping difficulty. We expect to apply a simple coordinate formula which is easy to obtain on generalized shape representation.

Three-dimensional to two-dimensional flattening is also a common application for surface geometric processing. Lévy *et al*. [LPRM02] proposed the as-conformal-as-possible method that flattens a triangle mesh to two dimensions for parameterization. Liu *et al*. [LZX*08] designed a hybrid flattening method to get more flexible results. All these methods inspire us to develop a generalized 4D-to-3D flattening algorithm.

3. Framework

This section gives a high level overview of our proposed framework. Our system takes as input a wide range of 3D textured solid models (Figure 1). For a tetrahedral mesh without texture, Takayama *et al*. [TOII08] proposed a method for interior solid texturing modelling. For volumetric datasets (like CT and MRI) with texture information only, we partition the given volumetric dataset using a uniform grid. Each vertex in the grid is associated with a 3D parameter (u, v, w). The original volume dataset now becomes the volume texture of the uniform grid. We further decompose each grid into several tetrahedra and convert the input to a 3D textured tetrahedral mesh, as shown in Figure 1 (bottom).

Now we can describe an arbitrary input by a uniform format. We define the input as a tetrahedral mesh $\mathbf{M} = \{\mathbf{T}, \mathbf{E}, \mathbf{V}\}$. $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$ denotes the set of tetrahedra, and $\{\mathbf{E}, \mathbf{V}\}$ denotes the set of edges and vertices. A mapping function ϕ maps vertices to the texture. In a discrete setting, each vertex $\mathbf{v}_i = (\mathbf{p}_i, \phi_i)$ includes two items: $\mathbf{p}$ denotes vertex's position (we use $\mathbf{p}^{3D} = (x, y, z)$ in three dimensions and $\mathbf{p}^{4D} = (x, y, z, h)$ in four dimensions). $\phi_i = (u, v, w)$ denotes a volumetric parameter corresponding to the volume texture. Our output is a new tetrahedral mesh $\mathbf{M}^{out}$ with updated $\mathbf{p}$ and ϕ for each vertex. Our framework includes the following steps.

(Step 1) Choosing ROI: The user makes an initial choice about ROIs. The shape/boundary of an ROI can be determined by a bounding sphere that encloses user's interested region, or, by a more accurate ROI's boundary. We could detect an accurate ROI's boundary through automatic boundary extraction operations (e.g. marching cube) or simple heuristic methods.

(Step 2) Magnification: In order to magnify the total volume in ROI, we generate a new 4D mesh $\mathbf{M}^{4D}$ based on the initial mesh.

- (2.1) We deform the original 3D tetrahedral patch inside the ROI in the 4D space, with the ROI boundary as constraints. So that no shape changes outside ROI's boundary, and the total volume within the boundary is magnified after this operation.
- (2.2) We recompute each vertex's parameter to remedy the shape distortion during magnification. To achieve this, we solve the volumetric harmonic function: $\Delta\phi^{12} = 0$, where ϕ^{12} is a texture transfer function $\mathbf{M}^{4D} \rightarrow \mathbf{M}$. Then for a vertex $\mathbf{v}_i = (\mathbf{p}_i^{4D}, \phi_i)$ in $\mathbf{M}^{4D}$, we update its parameter as: $\phi_i = \phi(\phi^{12}(\mathbf{p}_i^{4D}))$, where ϕ is the parameter on the original 3D mesh $\mathbf{M}$.

(Step 3) Flattening: In Step 2 we have already magnified $\mathbf{M}$ to $\mathbf{M}^{4D}$. In order to visualize $\mathbf{M}^{4D}$, it is necessary to flatten $\mathbf{M}^{4D}$ back into a 3D mesh as the final output $\mathbf{M}^{out}$ and preserve the magnification effect. We use a 4×3 rotation matrix to rotate each 4D tetrahedron $\mathbf{t}_i^{4D}$ back to a 'flattened' 3D tetrahedron $\mathbf{t}_i^F$. Then we stitch all separate tetrahedra together as the sole mesh $\mathbf{M}^{out}$, and keep each tetrahedron's shape to similar to $\mathbf{t}_i^F$ after stitching. We can execute this step iteratively until getting a visually promising result.

- (3.1) We initially guess a 3D tetrahedral mesh (e.g. from the last iteration's result, or by simple projection from $\mathbf{M}^{4D}$ in the first iteration). By comparing between the 'guess' tetrahedron $\mathbf{t}_i^{3D}$ in $\mathbf{M}^{3D}$ and rotation-generated 'flattened' tetrahedron $\mathbf{t}_i^F$, we can compute a 3×3 Jacobian matrix $\mathbf{J}_i$ between two corresponding tetrahedra. Then we can extract from $\mathbf{J}_i$ a stretching-free/rotation-only matrix $\mathbf{R}_i$.

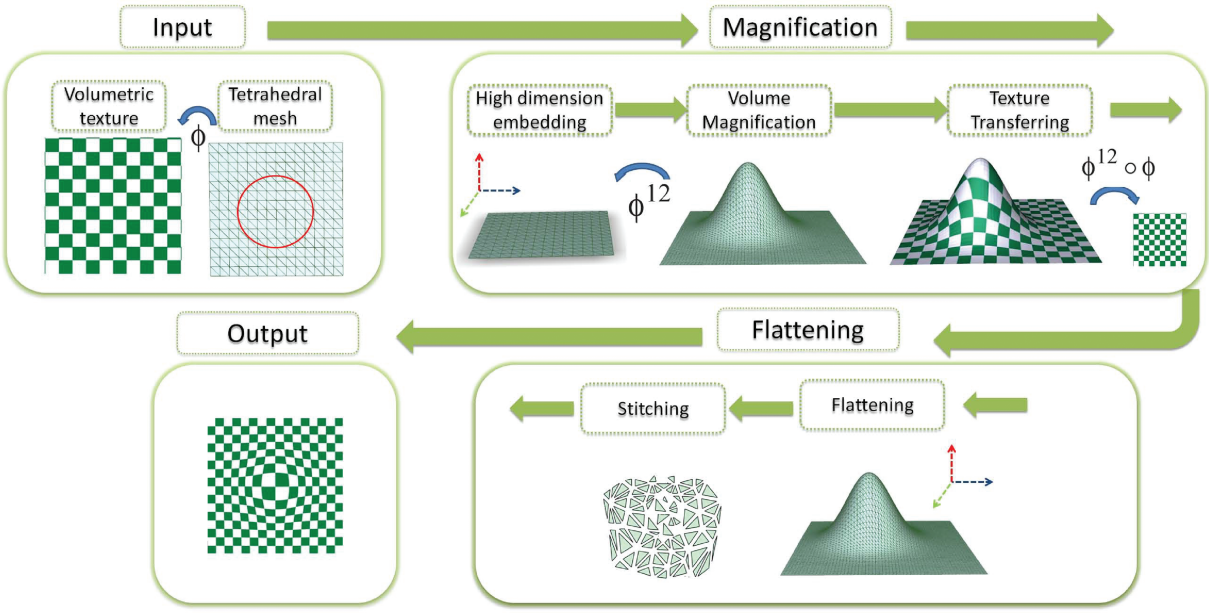


Figure 2: *Illustration of the framework. Because it is impossible to visualize 4D space, we use an image, a planar triangle mesh and a 3D triangle mesh to represent a volumetric dataset, a 3D tetrahedral mesh and a 4D tetrahedral mesh. After pre-processing, the input is a tetrahedral mesh with a volumetric dataset as the texture. The tetrahedral mesh is first embedded into a high-dimensional space and we magnify the total volume in an ROI through the additional dimension. We solve the harmonic function to recompute the mapping and transfer the texture to the new 4D tetrahedral mesh. Finally, we flatten the 4D tetrahedral mesh back into 3D for visualization.*

(3.2) We solve the linear optimization equation to determine every vertex's position in $\mathbf{M}^{\text{out}}$ such that, in the resulting mesh, the Jacobian matrix between the resulting tetrahedron and the 'guess' tetrahedron approximates $\mathbf{R}_i$.

Figure 2 shows our framework in a step-by-step fashion. Since it is extremely difficult to visualize the 3D-to-4D deformation in an intuitive way, we utilize 2D-to-3D deformation to simply illustrate the entire framework: 2D image and triangle mesh to mimic volume dataset and tetrahedral mesh, and deformed 3D triangle mesh to mimic a 4D tetrahedral mesh.

4. 3D-to-4D Magnification

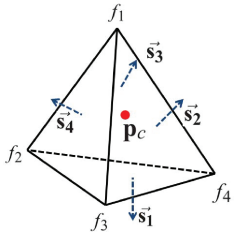
In order to magnify in 4D space, we first extend the input $\mathbf{M}$ by embedding it into 4D space. For each vertex with a 3D position $\mathbf{p}^{3D} = (x, y, z)$, we expand it to $\mathbf{p}^{4D} = (x, y, z, h)$, where the additional height $h = 0$. We can imagine this operation in the 2D layout as pulling a 2D plane from 2D to a real 3D world with shape unchanged (still a 2D plane but embedded in a 3D world after pulling).

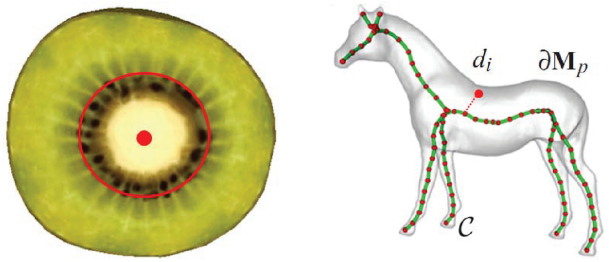
ROI magnification. Now we start to magnify ROIs. ROI is a region in the volume. Each ROI encloses a mesh patch $\mathbf{M}_p$ and we use $\partial\mathbf{M}_p$ to represent the boundary of patch $\mathbf{M}_p$. To magnify the ROI's volume, we seek a solution that could stretch all vertices inside $\mathbf{M}_p$ to new positions while keeping other vertices unchanged.

In most practical focus + context visualization applications, the user only chooses a general approximate region via simple user sketch or basic geometric primitives (like the region within a drawn sphere), enclosing both mesh segment and nearby context space as a reasonable proxy. In our system, we use a sphere to enclose the focus region and simulate lens in most applications. The choice of sphere lens is natural and humans are more accustomed to it with better visual understanding compared with other geometric primitives. In practice, we first visually choose a general approximate region, then we pick the centre $\mathbf{c}$ of this region as the centre of sphere associated with radius r, and r must be large enough to enclose the entire ROI.

After setting the lens, we magnify its volume by moving each vertex to a new position along the fourth dimension. As shown in Figure 2, we use a gaussian function to compute h_i in each $\mathbf{p}_i^{4D}$ because the shape changing in such case is not severe but smooth. For each vertex we compute $h_i = \mathbf{g}(1 - \frac{d_i}{r})h_0$, where d_i denotes the distance to the sphere centre $\mathbf{c}$, $\mathbf{g}(x)$ denotes a standard gaussian function e^{-x^2} and h_0 is a user input to scale the magnification. As an alternative solution, we can also use a standard 4D sphere instead of gaussian function to accommodate user's visual preference: $h_i = \sqrt{r^2 - d_i^2}$.

In some applications, the user may seek for a lens with an arbitrary shape. For example, a focus object extracted from the volume may have complex shape or high genus boundary and the user prefers to use this exact boundary to be the lens (like Figure 3 b). To achieve this, we can generate a central skeleton-like curved path $\mathcal{C}$ (e.g. [ATC*08]) and get the medial axis transform for every point on the object boundary. Each vertex $\mathbf{v}_i$





(a) (b)

Figure 3: *Two ways of lens shape design: (a) We can use a 3D sphere, with a centre* **c** *(red point), to enclose the entire ROI. The radius is r. (b) For an arbitrary shape lens like an extracted object's boundary (horse) from the volume, its medial axis can assist us to generate the lens. Each vertex inside the ROI associates a distance value d_i with the axis.*

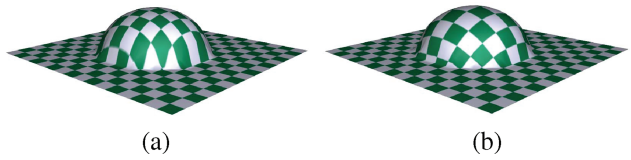
inside the lens associates the shortest distance d_i with the axis path $\mathcal{C}$. Now again we can use gaussian function to compute h_i for each vertex: $h_i = \mathbf{g}(1 - \frac{d_i}{d_m})h_0$, where d_m is the maximum distance value.

Large-scale magnification may stretch/shear the tetrahedron and sabotage the mesh quality. To solve this, we need to subdivide the highly stretched tetrahedron and compute the locations and parameters $(\mathbf{p}, \phi)$ for newly inserted vertices. We utilize barycentric coordinates and linear interpolation to interpolate new positions and parameters. For a point $\mathbf{p}_c$ inside a tetrahedron, its barycentric coordinate is:

$$f_c = \sum_i^4 \lambda_i f_i, \lambda_i = \frac{1}{3}\frac{< \mathbf{p}_c, \vec{\mathbf{s}}_\mathbf{i} >}{V}, \quad (1)$$

where V is the volume and $\vec{\mathbf{s}}_\mathbf{i} = A_i\vec{\mathbf{n}}_\mathbf{i}$. A_i indicates the area of one tetrahedron's face triangle (and each tetrahedron has four face triangles). Using the barycentric coordinates, we can keep the shape unchanged before and after adding vertices. Although the texture interpolation may not be optimal under this strategy, we compensate it by modifying the texture coordinates in the following texture transfer step.

Texture transfer. The tetrahedral mesh in the focus region has already been magnified after the magnification step. The following texture transfer step is necessary because after the above-mentioned magnification step, the tetrahedron in the focus region has already been significantly deformed to a different shape, thus still using the unchanged coordinates to map and interpolate the texture will inevitably cause angle distortion. Texture re-mapping can preserve the original texture shape after deformation. Figure 4 uses a surface in 3D example to illustrate the necessity of texture transfer (since we are unable to visualize the real volume in 4D example). Direct magnification without texture transfer (left) produces severe distortion effect for the context region, and texture transfer step can remedy this distortion (right).



(a) (b)

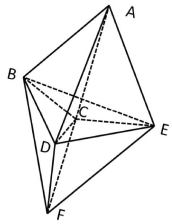
Figure 4: *Texture transfer in 2D layout to illustrate the effectiveness of texture transfer. (a) Direct magnification without texture transfer. (b) The result after texture transfer.*

The objective of this step is to texture the new mesh using the original texture, while preserving the initial texture shape. We have the tetrahedral mesh $\mathbf{M} \in R^3$ and $\mathbf{M}^{4D} \in R^4$ before and after the magnification. To transfer the texture from $\mathbf{M}$ (with the texture function ϕ) to $\mathbf{M}^{4D}$, it is desirable to construct a function ϕ^{12} : $\mathbf{M}^{4D} \rightarrow \mathbf{M}$, that maps the entire space of $\mathbf{M}^{4D}$ onto $\mathbf{M}$. Then we can describe the transferred texture mapping function on $\mathbf{M}^{4D}$ as $\phi \circ \phi^{12}$.

This function can be solved from the following harmonic function to minimize the mapping distortion: $\Delta\phi^{12} = 0$, where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$. We use discrete piece-wise coordinates to solve it numerically.

(1) In $\mathbf{M}^{4D}$, we use each vertex's original 3D position as the initial parameter $\phi_i = (u_i, v_i, w_i) = (x_i, y_i, z_i)$.
(2) We iteratively update each vertex's parameter $(u_i, v_i, w_i) = \sum_{Ng(\mathbf{v}_i)} \omega_{ij}(u_j, v_j, w_j)$, where $Ng(\mathbf{v}_i)$ is the one-ring neighbour of $\mathbf{v}_i$, (u_j, v_j, w_j) is every neighbour's parameter, ω_{ij} is the local coordinate associated with each neighbour. We keep parameters unchanged on the volume boundary vertices, as Dirichlet boundary conditions.
(3) ϕ^{12} now maps vertex $\mathbf{v}_i$ to one point location (u_i, v_i, w_i) on $\mathbf{M}$. Now we assign the texture parameter on this point in $\mathbf{M}$ to $\mathbf{v}_i$.

Volumetric local coordinates. We use cotangent coordinate here because it is a robust coordinate system and widely used on triangle mesh processing thus we generalize it from triangle mesh to volume ones. Note that, there are generally more than two tetrahedra sharing the same edge. Suppose for edge E_{uv}, it is shared by n tetrahedra thus it is corresponding to n dihedral angles, $\theta_i, i = 1, \ldots, n$, we define the string energy as $k_{u,v} = \sum_{i=1}^n \cot\theta_i$. Then for a vertex $\mathbf{v}_i$, we express its one-ring neighbour's local coordinate as $\omega_{ij} = \frac{k_{i,j}}{\sum_{Ng(i)}^j k_{i,j}}$. String energy k_{uv} involves computing the dihedral angles between two faces. We compute a dihedral angle in three dimensions as follows. As shown in the wrapped illustrative figure, we can compute the cosine of the dihedral angle between two opposite faces $\triangle ABD$ and $\triangle CDB$ as the following multiplicative term (up to the product of the norm of these vectors):



$$(\vec{AB} \wedge \vec{AD}) \cdot (\vec{CD} \wedge \vec{CB}). \quad (2)$$

However, in our 4D space $\mathbf{M}^{4D}$ this formula is not suitable for computing. It turns out that in 4D space, cross product operator '$\wedge$' requires three vectors rather than just two. To avoid using $\wedge$, we can use Lagrange's identity to compute the above formula:

$$(s \cdot u)(t \cdot v) - (s \cdot v)(t \cdot u) = (s \wedge t) \cdot (u \wedge v). \tag{3}$$

Now we can compute the cosine of the dihedral angle with the following updated formula:

$$(\vec{AB} \cdot \vec{CD})(\vec{AD} \cdot \vec{CB}) - (\vec{AB} \cdot \vec{CB})(\vec{AD} \cdot \vec{CD}). \tag{4}$$

5. Flattening

After the above step, we have already magnified the volume of ROI in a 4D mesh $\mathbf{M}^{4D}$. However, we have to flatten it back to 3D space for visualization and other applications. The key challenge in this step is to preserve every magnified tetrahedron's volume/shape during flattening. Inspired by 3D techniques like [SA07], [LZX*08], we devise a two-step algorithm to handle 4D flattening. We first rotate each 4D tetrahedron $\mathbf{t}_i^{4D}$ individually back to 3D space as the 3D tetrahedron (without changing shape except rotation). We denote this 'flattened' 3D tetrahedron $\mathbf{t}_i^F$. Note that every tetrahedron is rotated back to three dimensions independently, thus all $\mathbf{t}_i^F$ are separate from each other in three dimensions without being glued together. The second stage includes stitching them together into one piece as the original tetrahedral mesh structure. During stitching we minimize the shape distortion such that the final tetrahedron in $\mathbf{M}_i^{\text{out}}$ preserves the shape of $\mathbf{t}_i^F$.

Algorithm 1 The flattening algorithm.

```
Input: Initial 4D mesh M^4D,
     threshold ϵ
Output: 3D mesh M^out
for all t_i^4D ∈ M^4D
    //Compute a flattened tetrahedron
    t_i^F = Flatten(t_i)
end for
M^0 = Initialize(M^4D)
k = 0, d =INF_MAX
while d > ϵ do
    for all t_i^k ∈ M_i^k do
     //Compute Jacobian matrix
     J_i = Jacobian(t_i^k, t_i^F)
     //Rotation-only matrix
     R_i = SVD(J_i)
    end for
    //Build and solve Eq. 9
    Assemble(L, R, V^F)
    V^k = SolveEquation(L, R, V^F)
    //Compute moving distance
    d = MaxDistance(M^{k-1}, M^k)
    k = k + 1
end while
M^out = M^k
Output: M^out
```

Rotating a 4D tetrahedron $\mathbf{t}_i^{4D}$ back to a 3D tetrahedron $\mathbf{t}_i^F$ is simple. The challenge lies at keeping its shape after stitching in the resulting mesh $\mathbf{M}_i^{\text{out}}$. Our system affords two iteratively computed phases to achieve this goal. To clearly describe the algorithm, we denote k as the current iteration, then $\mathbf{M}^k$, $\mathbf{v}_i^k$, $\mathbf{t}_i^k$ as the tetrahedral mesh, a vertex and a tetrahedron in the k-th iteration, respectively. Note that $\mathbf{M}^k$ always keeps the same mesh structure as the input mesh $\mathbf{M}$. Initially, we generate the mesh $\mathbf{M}^0$ in the first iteration by removing the fourth dimension from every vertex in $\mathbf{M}^{4D}$: For a vertex with $\mathbf{p}_i^{4D} = (x_i, y_i, z_i, h_i)$ in $\mathbf{M}^{4D}$, we initialize its position in $\mathbf{M}^0$ as $\mathbf{p}_i^{3D} = (x_i, y_i, z_i)$.

In the first phase we compute the Jacobian deformation matrix for each tetrahedron $\mathbf{t}_i^k$. The matrix represents the transformation from the localized flattened tetrahedron $\mathbf{t}_i^F$ to its counterpart $\mathbf{t}_i^k$. We represent this transformation as a 3×3 matrix $\mathbf{J}_i$. Generalized from [HHN88], we can compute this Jacobian matrix as:

$$\mathbf{J}\left(\mathbf{t}_i^k\right) = \sum_{i=1}^{6} \vec{\mathbf{e}}_i^{\,k} \left(\vec{\mathbf{e}}_i^{\,F}\right)^T, \tag{5}$$

where $\vec{\mathbf{e}}_i^{\,k}$ and $(\vec{\mathbf{e}}_i^{\,F})^T$ are the corresponding edges between $\mathbf{t}^k$ and $\mathbf{t}^F$ (totally there are six pairs of edges for every tetrahedron). This matrix measures two tetrahedral deformation on two factors: rotation and scaling. Our goal is to preserve the shape of each tetrahedron thus we allow a rotation-only matrix, which can be decomposed separately by singular value decomposition of $\mathbf{J}$:

$$\mathbf{J}\left(\mathbf{t}_i^k\right) = \mathbf{U}\Sigma\mathbf{V}^T, \quad \mathbf{R}_i = \mathbf{U}\mathbf{V}^T, \tag{6}$$

where $\mathbf{R}_i$ is the rotation-only matrix.

Now in the second phase, we can update the position of each vertex by minimizing the following energy:

$$\mathbf{E}^k = \sum_{i}^{|\mathbf{T}|} \sum_{j=1}^{6} \kappa_{ij} ||\vec{\mathbf{e}}_{i\,j}^{\,t} - \mathbf{R}_i \vec{\mathbf{e}}_{i\,j}^{\,F}||^2, \tag{7}$$

where $\mathbf{T}$ is the set of all tetrahedra and $|\mathbf{T}|$ denotes the total number, $\vec{\mathbf{e}}_{i\,j}^{\,k}$, $\vec{\mathbf{e}}_{i\,j}^{\,F}$ are six edges on the tetrahedron $\mathbf{t}_i^k$ and $\mathbf{t}_i^F$, κ_{ij} is the weight associated with the edge. Now we rewrite the function in terms of every edge vector:

$$\mathbf{E}^k = \sum_{m,n} \kappa_{mn} || \left(\mathbf{v}_m^k - \mathbf{v}_n^k\right) - \mathbf{R}_l \left(\mathbf{v}_m^F - \mathbf{v}_n^F\right) ||^2, \tag{8}$$

where we use $\mathbf{v}_m^k - \mathbf{v}_n^k$ to represent every edge in Equation (7), $\mathbf{R}_l$ and κ_{mn} are the rotation-only matrix and weight of the tetrahedron $\mathbf{t}_l$ which the edge $(\mathbf{v}_m^k, \mathbf{v}_n^k)$ belongs to. Note that an edge $\mathbf{v}_m^k - \mathbf{v}_n^k$ may appear multiple times if it is shared by more than one tetrahedron, and thus we use different $\mathbf{R}_l$ when the edge appears more than once. Setting the gradient to zero, we obtain the following linear equation:

$$\mathbf{L}(\mathbf{V}^k)^T = \mathbf{R}\mathbf{L}(\mathbf{V}^F)^T, \tag{9}$$

where the matrix $\mathbf{L}$ represents the edge relationship of vertices (weighted by κ_{mn}) in Equation (8). The matrix $\mathbf{R}$ includes all local matrix $\mathbf{R}_l$, $\mathbf{V}^k$ and $\mathbf{V}^F$ are vectors including all vertices' positions on $\mathbf{M}^k$ and $\mathbf{M}^F$. $\mathbf{V}^k$ is the only unknown vector here and solving this equation gives rise to the positions of all vertices in $\mathbf{V}^k$.

After updating the positions, we compute the moving distance for each vertex between $\mathbf{M}^{k-1}$ and $\mathbf{M}^{k}$. The distance is normalized to the diagonal length of the volume. We record the maximum moving distance among all vertices, and the iteration loop stops if this distance is smaller than the threshold. We set the threshold to be 10^{-4}. In practice for all experimental results our algorithm converges in at most two iterations. The reason of this fast convergence is that our tetrahedral mesh is very simple: just a volume as an R^3 plane with a simple magnified function in the middle.

Weights. The choice of weight κ_{mn} in Equation (8) depends on the importance of a tetrahedron. From the cognitive perspective, tetrahedra around the ROI centre are more sensitive. Also a tetrahedron with large volume should have a higher weight than the one with small volume, because the distortion on a large tetrahedron is more visually confusing. For each edge, we design the weight as $(1+h)Vk_{u,v}$, where V is the average volume of connected tetrahedra, h is the averaged height (h-values) and $k_{u,v}$ is the string energy computed during texture transfer.

Boundary constraints. For a solid textured model, it is necessary to keep the boundary shape. For a volumetric dataset, the user also prefers to get a resulting shape with an original square boundary. Therefore, we keep the position of every boundary vertex unchanged during all iterations.

6. Experimental Results and Discussions

Our system can effectively provide magnification information to the user, allowing the user to get detailed focal region while maintaining the integral perception of the model. The results shown in the following figures demonstrate the power of our techniques. Our experimental results are implemented on a 3 GHz Pentium-IV PC with 4 Giga RAM. We utilize both magnification functions like sphere for Figures 5 to 7 and gaussian function for other figures.

We test our system on both solid textured models and volumetric datasets. From Figure 5 to Figure 6, we test various solid textured models such as watermelon and kiwi. Figure 5 demonstrates one important application using our focus + context magnification. The figure shows that more seeds appear after magnification. Also, the distribution of seeds (i.e. their relative positions between seeds) is preserved. Preserving particle distribution and relative positions during magnifying has many potential applications in experiment-driven science and engineering (e.g. structural biology, game design). Our focus + context magnification provides an effective magnification lens for this category of applications.

Figure 5: *The tetrahedral mesh of watermelon.*

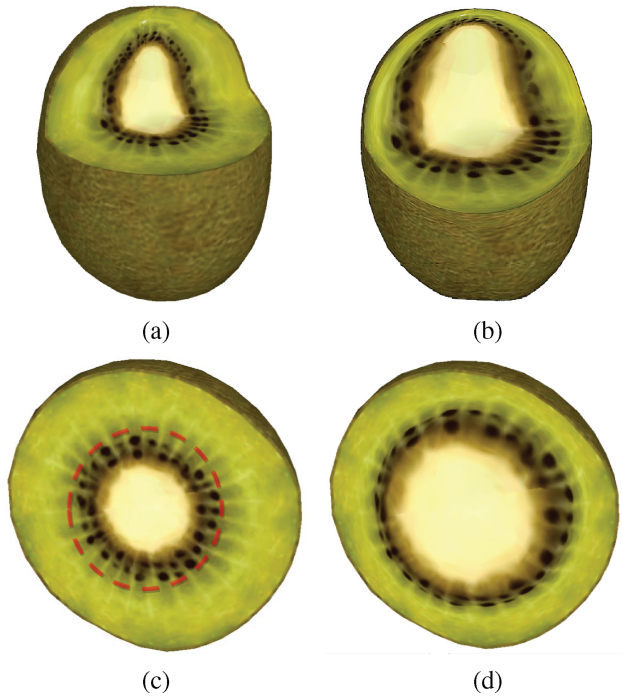


Figure 6: *The tetrahedral mesh of kiwi.*

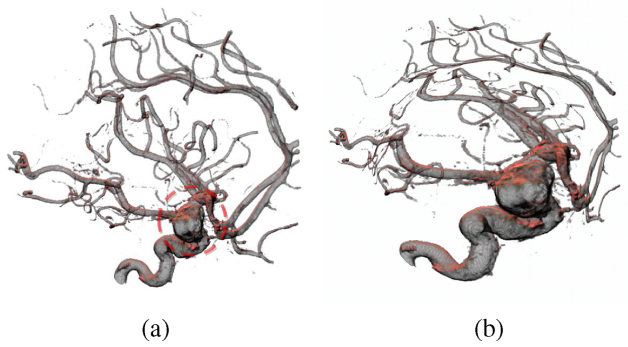


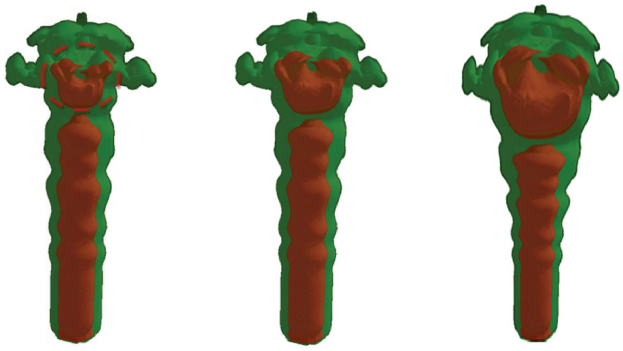
Figure 7: *The volumetric aneurism dataset.*

Figure 6 shows another example. Compared with [WWLM11], in which the sphere-like shape is severely distorted (e.g. the brain model is severely distorted to an irregular heart-like model), our lens successfully keeps the structure of kiwi core still as the spherical shape, and the shape of context region is also unchanged.

From Figure 7 to Figure 9, we test several volumetric dataset examples: aneurism, bonsai and fuel. In these tests, we magnified different shapes like tumour in Figure 7, trunk in Figure 8 and irregular air head in Figure 9. All experimental results clearly demonstrate that our framework can keep the prominent global

(a) (b)

Figure 8: *The volumetric bonsai tree dataset with magnified trunk.*



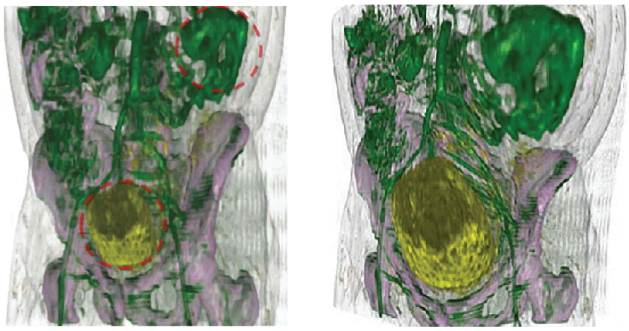
(a) (b) (c)

Figure 9: *The volumetric fuel dataset with magnified head.*

shape and the context region unchanged for viewers' easy recognition. Meanwhile, in Figure 7 we demonstrate an application on structure-aware visualization using a model with many branches (in many practical applications the input model includes many branches). We magnify the tumour model while long branches (thinner vessels) are preserved without occlusion or relative position distortion. This example shows that our method could be of great value to structure-critical applications (e.g. oil pipeline optimization and detection, indoor routing and planning). Figure 7(b) shows that we can observe the wrinkles on the tumour after magnification.

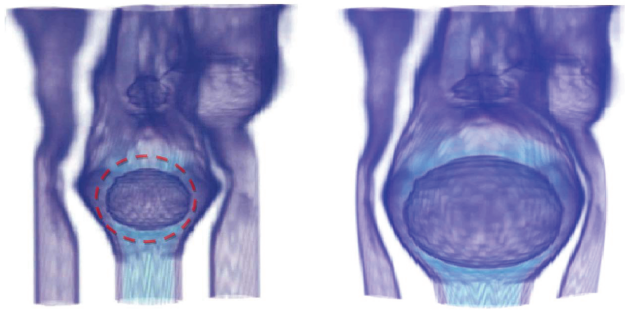
Figure 8 demonstrates the application of arbitrary shape lens. In most of our examples we use the standard sphere shape lens. However, as we discussed in Section 4, we can also generate arbitrary lens shape. We first extract a 2D rectangle-like shape trunk boundary separately on each image layer along 3D input's z-axis. Then we can reconstruct the 3D ROI boundary from these 2D boundaries and generate the medial axis centre line. The trunk is magnified and the shape is well preserved. In Figure 9 we show that we can modify the magnification ratio. This function is achieved through changing the height of our gaussian function.

In Figures 10 and 11 we demonstrate more applications such as medical and physics experiment visualization with complicated models. We magnify the multiple parts in Figure 10 and the resulting



(a) (b)

Figure 10: *The volumetric body dataset with multiple ROIs.*



(a) (b)

Figure 11: *The volumetric smoke dataset.*

Table 1: *Statistics of various test examples: # Tex, # pixels in the texture; # V, # of vertices; Distortion, average distortions by all vertices.*

Model	# Tex	# V	Time	Distortion
Melon	64^3	600	1.5 s	0.05
Kiwi	64^3	880	2.1 s	0.08
Aneurism	256^3	30^3	315 s	0.07
Bonsai	256^3	40×20^2	127 s	0.03
Fuel	64^3	25^3	110 s	0.08
Body	128^3	30^3	275 s	0.03
Smoke	256^3	30^3	340 s	0.04

model preserves the context region very well. The user (doctor) can easily recognize each surrounding part (pelvis, artery, etc.) without any difficulty. This advantage enables the doctor to obtain the accurate information and avoid mis-diagnosis. In Figure 11, we magnify the smoke obstacle while we can still recognize the details like shape and number of surrounding flows.

Performance. The performance of our framework does not depend on the size of texture/volumetric datasets but the size of vertices in the input tetrahedral mesh. The statistics of examples are shown in Table 1. About computational time, in practice, we can interactively use a sparse low-resolution cube/tetrahedral volume, like in [WWLM11], to accelerate the computation and get a fast result. Furthermore, we can pre-compute magnification and flattening on

pre-designed mesh and later use it on different volumes by just changing textures of the mesh.

Compared with other optical-/voxel-/resizing-based methods, our geometry-based method has the advantage that we can quantify the local distortion by computing mesh angle distortion, instead of just displaying visual effects. In the conventional lens design techniques, the user can only recognize the distortion through observation because of lacking an accurate measurement method. By comparison, our focus + context lens defines two categories of distortions: the local distortion and global distortion. We define the local distortion as the angle distortion in each tetrahedron. This metric can be quantified by computing the ratio of the single values σ_1 and σ_2 from the Jacobian matrix $\mathbf{J}$ (the metric is normalized by the diagonal length of the whole cube grid volume).

Generally, the flattening step may cause self-intersection result if the minimization solver in the flattening algorithm leads to a local minimum solution. However, our system can avoid this self-intersection, because the input (magnified 3D plane) we use is very simple (here 'simple' means its shape is very close to the final flattened shape). To theoretically illustrate its robustness on how to avoid self-intersection, we shall notice that our flattening algorithm is a 3D generalization from the surface method [SA07, LZX*08], which is originally designed to handle very complex and/or high genus surface model input with no self-intersecting triangle in the output. In practice this method can effectively handle a model with very complex shape without self-intersection. Compared with these complex surface models, our model's geometry and topology structure is rather simple: a flattened R^3 plane with a simple gaussian function in the middle. This simple structure means the deformation is rather slight from this simple input to a flattened output. In our experimental results, the self-intersection does not visually appear. Consequently, degeneration prevention is not practically necessary for our mesh, thus our system does not need to provide more mechanism to prevent self-intersection.

Comparisons. Currently most of magnification lens design focuses on 2D image visualization only. Recently, Wang *et al*. in [WWLM11] introduced a data reduction method which can achieve magnification effect. Compared with [WWLM11], our method's most important advantage is that: we can preserve both the focus region's shape and the nearby/global shape. Although the method in [WWLM11] can preserve the shape surrounding the focus region, it is incapable of preserving the nearby transition region (e.g. context), especially the global shape. These phenomena appear in the examples of [WWLM11] and show their method's major limitation. For example, in [WWLM11] Figure 1, column 2, in order to magnify the focus region, the entire brain model (i.e. the global shape) is distorted significantly: from an original sphere-like shape to an irregular heart-like shape. In another focus + context visualization example in [WWLM11] Figure 8, the contour of skull is severely deformed. Such severe distortion of the global shape may cause misunderstanding/mis-diagnosis [WQK06]. By comparison, our technique preserves the context region and global shape much better than [WWLM11]. For example, our method can keep the sphere boundary of watermelon and kiwi unchanged after magnification (Figures 5 and 6). Therefore, our method with an improved context region/global shape-preserving capability could be more useful in the relevant applications. We shall notice that in [WWLM11]'s example, multiple ROIs are magnified within the brain model. We also magnify the multiple ROIs in Figure 10 to demonstrate that our method still can well preserve the nearby region when magnifying multiple ROIs. Also, the key factor to the quality of shape preserving is total size of ROIs rather than number of ROIs, because texture transfer and flattening optimization are solved globally without using ROIs' boundaries as constraints. The total size of ROIs in our tests always takes a large percentage of the input datasets, to demonstrate our method's efficiency.

Another comparison is on distortion measuring and quantifying. The distortion mechanism in [WWLM11] is highly arbitrary, determined by weighted cube grid magnification. Our method is geometry based and generalized from the surface conformal parameterization technique, thus we can measure the local angle distortion much better from the perception's point of view. Angle-oriented shape perseveration and distortion minimization are more perceptually pleasing than using cube grid in [WWLM11]. Their cube resolution is very coarse with hundreds of voxels inside each cube. The linear interpolation of these voxels after cube grid deformation will cause additional angle distortion. Therefore, the cube grid distortion metric is always inaccurate. Our system can visualize the distortion not only through visual display but also quantifying such effect by computing angle distortion in a more accurate way (which is the ratio of two singular values from Jacobian matrix).

We also compare our method with another focus + context technique [WLT08]. We shall notice that our system handles much more complicated scenarios than those in [WLT08]. The input in [WLT08] is only surface boundary model, so it has no interior or nearby information to display or magnify (all nearby context regions are empty 3D space). Consequently, [WLT08]'s system can hide severe distortions in the empty context region without any visual information (since it is invisible). By comparison, our input is 3D solid model or volume with multiple materials/tissues, both inside the focus region and outside such region. When we magnify a focus region inside our model, all nearby context regions should avoid distortion because they also contain important tissue, material and shape information. By comparison, our geometry-based method can accommodate more complicated models with well-preserved magnification results for interior and exterior regions.

Since our lens is geometry based, it can effectively obtain a better global distortion minimization even on surface mesh when only compared with [WLT08]. We can simply modify our framework to support surface-only triangle mesh: we use a polycube to cover the whole input mesh and then magnify the polycube. Figure 12 compares our method with the result in [WLT08]. After setting the user-selected focus region (red circle in Figure 12a), the magnification result generated by Wang's method preserves structure/shape in the focus area, but severely affects the context region (e.g. the upper body, red circle in Figure 12b) and introduces visual artifacts, like the distorted proportion of body. By comparison, our technique keeps upper/lower body proportion without obvious shape confusion for easy object recognition (red circle in Figure 12c).

Our lens is also similar to the mesh-editing–based method (which is equivalent to magnifying the surface boundary first and then

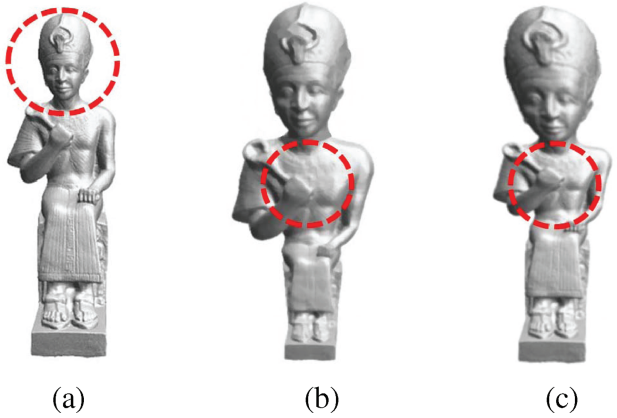


Figure 12: *Comparison between Wang's method. (a) Input and the focus region (red circle). (b) [WLT08]'s method (courtesy of [WLT08]) and its resulting context region (red circle). (c) Our method and the resulting context region (red circle).*

interpolating the interior texture). However mesh-editing techniques are not suitable for the focus + context visualization application because they focus on totally different input and task. First, mesh editing requires users to operate on an exact mesh boundary segment. However, in focus + context visualization applications, the desired regions cannot be easily detected, extracted and described as the triangle mesh model. For example, boundary extraction is extremely difficult for most volumetric/medical datasets. In most practical focus + context visualizations, the user only chooses a general/approximate region via simple user sketch and/or basic geometric primitives (like the region in a drawn circle), enclosing both mesh segment and nearby context space as a reasonable proxy. Second, mesh editing only attempts to preserve the shape of focus region around mesh boundary during magnification. The nearby context region and global shapes will be severely distorted without consideration (in most cases, these regions are just empty space in a typical mesh-editing task). Finally, mesh editing only focuses on surface mesh's shape, thus for interior textures/tissues, we still need to design a shape-preserving interpolation technique to preserve the shape after boundary deformation. In Table 2, we compare our method with [WWLM11, [WLT08] and mesh-editing methods. The table clearly shows that our method is a more powerful tool for volume data focus + context visualization.

Limitations. At present, our framework still lacks of mechanism to handle some key features like long straight edges in the input volume, especially when sharp features are on the boundary of the context region. For most visualization applications, this drawback may not be obvious. However, for visualization involving manufactured objects in traditional CAD (like crack analysis), this distortion may cause severe difficulty during object exploration. We shall study how to design better algorithms to support this type of applications to keep meaningful sharp features like cracks unchanged during magnification.

Also, our framework cannot guarantee to uniformly magnify multiple chosen regions or accurately modify different regions' magnification ratio. This requirement is important in many applications like virtual surgery analysis, where all focused polyps should be magnified in the equal ratio in order to avoid mis-diagnosis. The user can also obtain flexible magnification results through controlling magnification ratio.

Table 2: *Comparison with various methods. We test their abilities in following aspects: Preserving the shape of focus region (Focus region); Preserving the shape around context region and/or global shape (Context region global shape); Supporting solid model and/or volume dataset (Solid texture); Quantifying local distortion (Distortion quantifying); Allowing simple sketch to choose ROI (Simple sketch input).*

Model	[WWLM11]	[WLT08]	Mesh editing	Ours
Focus region	Yes	Yes	Interpolation needed	Yes
Context region global shape	No	No	Empty space	Yes
Solid texture	Yes	No	No	Yes
Distortion quantifying	No	No	Yes	Yes
Simple sketch input	Yes	Yes	No	Yes

Theoretically, our method can achieve a large-scale magnification result. However in practice, extreme magnification may still cause distortion. Also, in some practice we choose to visualize the volume interior region by cutting the model and observe the cross-section on the screen. The magnification deforms the whole model and the original cross-section may not be on the same cutting plane anymore, forcing the observer to modify or re-cut the cross-section.

7. Conclusion and Future Work

In this paper, we have developed novel techniques towards designing magnification lens for volumetric datasets. Specifically, we start from the input of a 3D tetrahedral mesh and magnify the ROIs in 4D space through the use of dimensional enhancement. Our geometry-centric methodology and the prototype system manifest that the 4D geometry greatly empowers the visualization techniques. Our approach is expected to transcend the traditional boundary of geometric modelling and is of benefit to data visualization and visual analysis.

Our framework outperforms other methods with many unique features. It offers users the immense power on shape distortion minimization and quantitative measurement. Compared with other methods, our system can preserve the shape not only around the focus region but also the surrounding context region and global shape. Also, it enables the user to draw either simple sketch (like drawing a sphere) or arbitrary shape as magnifiers to effectively display the entire ROIs. Our system affords a wide spectrum of 3D input ranging from volume datasets to solid textured models.

In near future, we continue to explore the utility of 4D geometric modelling/processing. At present, our framework still lacks of mechanism to handle sharp features like long straight lines in the input volume. We shall study how to design better algorithms to keep meaningful sharp features unchanged during magnification.

The whole algorithm could also be accelerated in parallel on GPU platform. Meanwhile, we also observe that the current scheme is capable of handling higher dimensional datasets, like solid textured models equipped with multiple vector fields/heterogenous materials. Four-dimensional models will provide more efficient representation for these datasets. We will also explore its applications on more generalized models like multivariate splines. For volume image reconstruction and vectorization like [LQ11], this technique could better preserve image features and thus we could utilize our method to improve reconstructed continuous representations for volume image processing and visualization.

Acknowledgements

This research is supported in part by National Natural Science Foundation of China (Grants No. 61190120, 61190121 and 61190125) and National Science Foundation of USA (Grants No. IIS-0949467, IIS-1047715 and IIS-1049448).

References

[ATC*08] Au O. K.-C., Tai C.-L., Chu H.-K., Cohen-Or D., Lee T.-Y.: Skeleton extraction by mesh contraction. *ACM Transactions on Graphics 27*, 3 (2008), 44:1–44:10.

[BSP*93] Bier E. A., Stone M. C., Pier K., Buxton W., Derose T. D.: Toolglass and magic lenses: The see-through interface. In *Proceedings of SIGGRAPH '93* (1993), pp. 73–80.

[CCF97] Carpendale M. S. T., Cowperthwaite D. J., Fracchia F. D.: Extending distortion viewing from 2D to 3D. *IEEE Comput. Graph. Appl. 17*, 4 (1997), 42–51.

[CS07] Correa C. D., Silver D.: Programmable shaders for deformation rendering. *ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (2007), *89*–96.

[FH05] Floater M., Hormann K.: Surface parameterization: A tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*. Springer, Berlin, Heidelberg, 2005.

[Flo03] Floater M.: Mean value coordinates. *Computer Aided Geometric Design 20*, 1 (2003), 19–27.

[Fur86] Furnas G.: Generalized fisheye views. *Human Factors in Computing Systems, CHI Conference Proceedings* (1986), pp. 16–23.

[HHN88] Horn B. K. P., Hilden H., Negahdaripour S.: Closed-form solution of absolute orientation using orthonormal matrices. *Journal of The Optical Society America 5*, 7 (1988), 1127–1135.

[HLS07] Hormann K., Lévy B., Sheffer A.: Mesh parameterization: Theory and practice. In *SIGGRAPH' 07* (San Diego, CA, USA, 2007), ACM SIGGRAPH' 07 Courses, ACM.

[JSW05] Ju T., Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. In *Proceedings of SIGGRAPH '05* (2005), pp. 561–566.

[KFCO*07] Kopf J., Fu C.-W., Cohen-Or D., Deussen O., Lischinski D., Wong T.-T.: Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 2:1–2:9.

[LGW*07] Li X., Guo X., Wang H., He Y., Gu X., Qin H.: Harmonic volumetric mapping for solid modeling applications. In *SPM '07: Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling* (2007), pp. 109–120.

[LHJ01] LaMar E., Hamann B., Joy K. I.: A magnification lens for interactive volume visualization. *Pacific Conference on Computer Graphics and Applications* (2001), pp. 223–232.

[LLCO08] Lipman Y., Levin D., Cohen-Or D.: Green coordinates. *ACM Transactions on Graphics 27* (2008), 78:1–78:10.

[LLWQ13] Li B., Li X., Wang K., Qin H.: Surface mesh to volumetric spline conversion with generalized poly-cubes. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1539–1551.

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH02* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 362–371.

[LQ11] Li B., Qin H.: Feature-aware reconstruction of volume data via trivariate splines. In *Proceedings of the 19th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2011)* (2011), pp. 49–54.

[LQ12] Li B., Qin H.: Component-aware tensor-product trivariate splines. *Computer & Graphics 36*, 5 (2012), 329–340.

[LZX*08] Liu L., Zhang L., Xu Y., Gotsman C., Gortler S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504.

[MCK] Martin T., Cohen E., Kirby R.: Volumetric parameterization and trivariate b-spline fitting using harmonic functions. *Computer Aided Geometric Design 26*, 6, 648–664.

[MDSB02] Meyer M., Desbrun M., Schröder P., Barr A.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Proceedings of Visualization and Mathematics* (Berlin, Germany, 2002), Springer, pp. 113–134.

[NBM*06] Nekrasovski D., Bodnar A., McGrenere J., Guimbretière F., Munzner T.: An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proceedings of CHI Conference on Human Factors in Computing Systems* (2006), pp. 11–20.

[PA08] Pietriga E., Appert C.: Sigma lenses: Focus-context transitions combining space, time and translucence. In *Proceedings of Conference on Human Factors in Computing Systems* (2008), pp. 1343–1352.

[PBA10] Pietriga E., Bau O., Appert C.: Representation-independent in-place magnification with sigma lenses. *IEEE*

Transactions on Visualization and Computer Graphics 16 (2010), 455–467.

[SA07] Sorkine O., Alexa M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (2007), pp. 109–116.

[SPR06] Sheffer A., Praun E., Rose K.: Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision 2*, 2 (2006), 105–171.

[TOII08] Takayama K., Okabe M., Ijiri T., Igarashi T.: Lapped solid textures: Filling a model with anisotropic textures. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 27*, 3 (2008), 1–9.

[Wac75] Wachspress E.: *A Rational Finite Element Basis*. Academic Press, New York, 1975.

[WGC*04] Wang Y., Gu X., Chan T.-F., Thompson P., Yau S.-T.: Volumetric harmonic brain mapping. In *ISBI'04: IEEE International Symposium on Biomedical Imaging: Macro to Nano* (2004), pp. 1275–1278.

[WLT08] Wang Y.-S., Lee T.-Y., Tai C.-L.: Focus+context visualization with distortion minimization. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 1731–1738.

[WQK06] Wei H., Qiu F., Kaufman A.: A pipeline for computer aided polyp detection. *IEEE Transactions on Visualization and Computer Graphics 12*, *5* (2006), 861–868.

[WTSL08] Wang Y.-S., Tai C.-L., Sorkine O., Lee T.-Y.: Optimized scale-and-stretch for image resizing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH ASIA) 27*, 5 (2008), 118:1–118:8.

[WWLM11] Wang Y.-S., Wang C., Lee T.-Y., Ma K.-L.: Feature-preserving volume data reduction and focus+context visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (2011), 171–181.

[ZLWK12] Zhao X., Li B., Wang L., Kaufman A.: Texture-guided volumetric deformation and visualization using 3d moving least squares. *Visual Computer 28*, 2 (2012), 193–204.

[ZZG*12] Zhao X., Zeng W., Gu X., Kaufman A., Xu W., Mueller K.: Conformal magnifier: A focus+context technique with local shape preservation. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1928–1941.