Interactive Volume Manipulation with Selective Rendering for Improved Visualization

Vikas Singh Deborah Silver

Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey
Piscataway, NJ 08855-0909
http://www.caip.rutgers.edu/vizlab.html

ABSTRACT

In this paper, we describe a methodology that will allow a viewer to select "geometrical" components of a volume and both manipulate these components and/or "highlight" them. A considerable number of volumetric datasets are now available and can be obtained from imaging techniques such as magnetic resonance imaging (MRI), cryogenic slicing, CT, or laser scans. In order to be able to better visualize these models, it is often required to repose them, e.g., remove occlusion, or to highlight various components. However, it may be still desirable to keep the rest of the volume present (and not to just cut it away) to maintain a focus+context view of the data. The methodology we propose allows the user to select regions of the data based upon the shape of the volumetric model. Once selected, the regions can be moved or highlighted via an alternate rendering. Some examples of alternate renderings include selective compression (higher resolution data is used to render the selected region), selective juxtaposition (replacing that selected component with data from another dataset), or selective transfer function determination. Both the selection and the rendering are done interactively to maximize their effectiveness. We show examples of the approach using a variety of volumetric datasets.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques; I.3.7 Three-Dimensional Graphics and Realism.

Keywords: Improved visualization, volume manipulation, IK (inverse-kinematics) skeleton, selective rendering, modeling, animation.

1 INTRODUCTION

Three dimensional volume datasets are ubiquitous in the medical, scientific and graphics domains. These datasets are the result of mathematical simulations, such as computational fluid dynamics, or are generated by observational equipments such as CT, MRI, ultrasound and laser/range scanner. There is a significant research in techniques to render these datasets quickly and realistically. Several APIs [27, 41, 42] and commodity hardware [10, 44] are available for rendering volumetric datasets. For scientific analysis and computer graphics applications, it is interesting to be able to manipulate these datasets interactively and accurately. Volume manipulation includes deformation, animation, smoothing, reshaping, and moving/removing parts of a volume.

Manipulation can also provide new views of volumetric models to facilitate better and improved visualization. It can be used as a teaching tool, to allow a student to discover new views, for dosimetry studies [43], volume graphics, virtual reality and virtual medicine. For example, in the original visible man dataset from NLM [33], the hands are positioned in front of the torso as can be seen in Figure 1 (a). Positioning a cutting plane between the hands and the torso would be very difficult and would cut out part of the arms. However, moving the hands away would be a "geometrical" type of manipulation that one may want to do. In Figure 1(b) an example of this type of manipulation is shown which thereby enabled better visualization. In this figure, the arms were moved by selecting the arms in the corresponding "IK skeleton" (shown below). Once selected, the arms can be manipulated, and/or can be rendered differently, i.e., they can be highlighted, made more transparent, or any of the improved rendering techniques could be added [7, 19, 30]. By allowing shape-based selections of the volume, i.e., pieces of the volume that correspond to different components, one can both manipulate and selectively render these components. Selective rendering refers to interactively selecting a portion of a volume for subsequent focused operation on it independent of the rest of the volume (context). To focus on the selected region one may choose from a variety of options such as rendering, compositing and transfer functions as discussed in later sections. This also provides the ability to create interactive Focus+Context visualization (see next section).

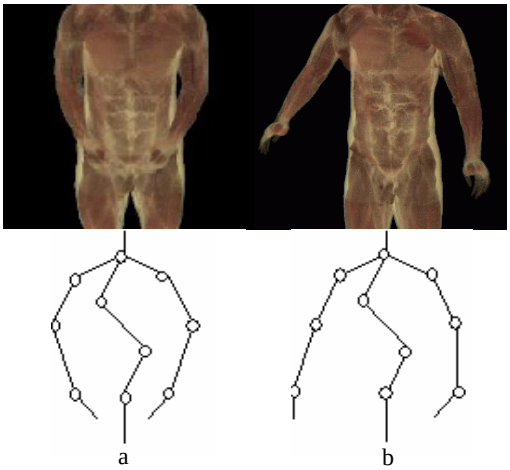


Figure 1. An example manipulation for removing occlusion. The corresponding IK skeletons are also shown. (a) original position of the hands, (b) the hands moved to show the torso

In the next section, we describe related research followed by our algorithm to manipulate and selectively render parts of a volume. Different examples are shown on various 3D datasets such as the visible human, a vascular tree, lobster and bonsai.

IEEE Symposium on Volume Visualization and Graphics 2004
October 11-12, Austin, Texas, USA
0-7803-8781-3/04/$20.00 ©2004 IEEE

2 RELATED WORK

There are two main areas of research related to the work proposed in this paper. These include volume manipulation and "selective" rendering. Manipulation of volumetric datasets is a challenging task. Most research in volume manipulation involves transforming object voxels or obtaining physically based models. The most popular techniques convert the 3D dataset into a polygon mesh for manipulation. Tools to manipulate polygonal meshes are mature and relatively easy-to-use. The polygons are then voxelized to create the 3D dataset in the new pose. However, this is potentially a lossy method, especially, if surfaces cannot be created for all interior regions. Volumetric models can also be manipulated with physically-based models. In [5], a system is presented using a volumetric mass spring model and an FEM model for animating volume objects. In [18], a 3D Chain Mail algorithm is used to propagate deformation along the coupled chain links through a volume. These sophisticated techniques require physical properties of the object like elasticity and mass, and are sometimes an over-kill for simple manipulations. In [25], a ray deflector technique is described for rendering deformations but is limited to visual deformations and the object is never directly deformed. Volume morphing [20] is a technique for targeted deformation and may not be always suitable in this case as the targeted volume is not available. A deformation-based technique for browsing volumetric data is presented in [31] and facilitates visualization of segmented layers, however, this is not applicable for the type of manipulation described in this application, i.e., specifying a logical part of the volume to manipulate. In [13, 14, 15, 16, 17, 36, 37] a volume animating system has been presented which allows users to specify an animation for a volume in a similar way as they would specify for polygonal models. This approach is shape-based and allows for more intuitive and logical manipulation, for instance, moving an arm of a man. This is the approach utilized in this paper, and it is described in more detail in section 3. In [37], we presented a system for interactively manipulating volumetric models. The method was based on computing a simple geometry for a volume based on its distance field, and then texture mapping the deformed geometry in graphics hardware. In this paper, we have extended the system to support different rendering modes, including texture based and point splatting.

Hardware based deformation and manipulation techniques have been presented in [6, 26, 35, 45]. These techniques include warping a polygonal hull [45] and/or warping the volume in texture space [35]. In [6] an extension to the OpenGL API is proposed to integrate free-form deformation into the OpenGL rendering pipeline. In order to utilize graphics hardware acceleration, the ray deflector approach in [25] has been extended in [26], by applying the deflectors on tessellated texture mapped polygons. Some of these techniques require costly pre-processing step such as in [45], or require considerable human input for manipulation such as [5], and the manipulation is restricted and non-intuitive. A recent commercial software package, Varipose, claims to reconfigure the visible male mesh including the internal tissue structures, although the actual methodology utilized is unclear as it is proprietary. This tool has been specifically designed for, and is only applicable to the visible human dataset. Moreover, mesh generation of the new pose may take hours depending on mesh size and repositioning.

Different techniques are available for visualizing new views of volumetric models. These techniques include the cutting planes, non-photorealistic rendering (e.g., [7, 30]) and visualizing segmented datasets (e.g. [19]). While cutting planes are very important for 3D visualization, they are sometimes hard to position accurately. In [19], an algorithm is presented to render segmented volumes on graphics hardware. Each segmented component of the volume can be rendered using a different rendering method, such direct volume rendering, iso-surfacing or non-photorealistic techniques, however, there is no way to distinguish different spatial components of a volume. The components are viewed as segmented regions. "Splitting" datasets for alternate rendering has also been utilized in [23], however, the approach is not interactive and uses ray casting for rendering. The method presented in this paper complements that approach and can be used to attain higher quality renderings. The main contribution of this paper is to provide an interactive technique to decompose a volume into meaningful logical regions based on the shape such that each region could be separately interacted with. The interaction with regions (or the whole volume) encapsulates both manipulation, and rendering. The user can select a region, manipulate and render it so that it becomes the "focus", while the rest of the volume is still maintained in context. This adds another layer to any rendering methodology, since both segmented and/or logical components can now be rendered selectively. Also, for sampled datasets, since the anatomy at certain regions may be better visualized with a different transfer function or resolution, the ability to interactively pick, maneuver and/or highlight those regions, will provide a very powerful tool for visualization.

3 THE VOLEDIT APPROACH TO IMPROVED VISUALIZATION

The goal of this paper is to present a new approach to creating improved and interactive volume visualization using *selective rendering*. It allows the user to interactively select portions of the data for manipulation and/or alternate rendering, thereby creating a focus+context [12, 29, 38] view of the data. Focus and context refers to visualizations that enable users to have the object (or part of the object) presented in detail while it is still embedded in its original context. In this work, we demonstrate how the user can interactively choose different regions of the data and choose the type of focus desired for those regions. The type of focus can include different transfer functions, different resolutions, and compositing.

The approach taken in this work is based upon the interactive manipulation methodology presented in [35, 37, 45] which utilizes a skeleton, which is a stick figure representation of the object, to allow the user to interact more effectively with the model. The skeleton is a reduced representation of the model that can easily be understood. In standard computer graphics, digital characters are animated using a skeleton (also called an IK-skeleton). The animator first creates a skeleton of the model and then binds the polygons to the skeleton. The skeleton can then be manipulated or animated to cause the corresponding movement in the model using key framing, inverse kinematics or motion capture. This process is followed in commercial animation packages such as Character Studio [1], or Maya [32].

Our volumetric equivalent process consists of three basic steps: skeletonize the volumetric objects (thin the volume and determine the bones and joints), choose bones for manipulation and/or selective rendering (interactively), and render the new view. An overview of this process is shown in Figure 2. Once selected, components are rendered within their context. Two different interactive rendering techniques are utilized and these are described in Section 3.2. Alternately, non-interactive rendering techniques can be applied [27, 41, 42]. In the following sections, we describe and illustrate the different steps in more detail. The

example datasets used include (a) three different versions of the Visible Human Dataset [33], all at different segmentations, with resolutions of `310x190x960`, `196x114x626`, and `586x340x1878`; (b) the lobster dataset [40] at a resolution of `256x256x64`; (c) the aneurism [40] is `256x256x256`; and (d) bonsai dataset [40] is `165x230x240`.

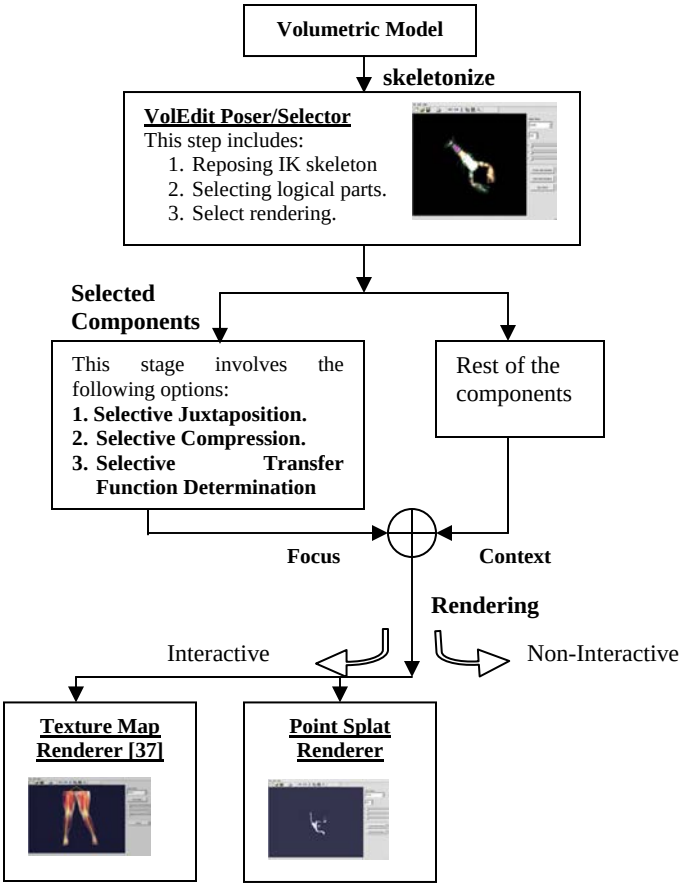


Figure 2. An overview of the manipulation and rendering process.

3.1 Skeleton Determination

The first step of the pipeline is to compute a skeleton from the volumetric object. A skeleton of a volumetric object is a useful shape abstraction that captures the essential topology of an object. Since the skeleton is a reduced representation, an intuitive method for shape deformation is to manipulate the skeleton to cause corresponding deformation in the volume. The skeleton can be extracted by using a variety of methods (see [14] for more information). In this paper the skeleton was determined using the method of [14], which is a thinning based algorithm, and [46] which is a potential field based algorithm. Once the skeleton has been obtained, the IK bones and joints are chosen. The joints can be chosen interactively, or semi-automatically. The voxel-to-bone correspondence map is automatically generated based on the proximity of each voxel to a bone and the shortest voxel-to-bone path. More details can be found in [14]. The advantage of [46] is that both the bones and joints are automatically determined, as are the voxel correspondences (which outer voxels are connected to which bone), however, [14] is faster.

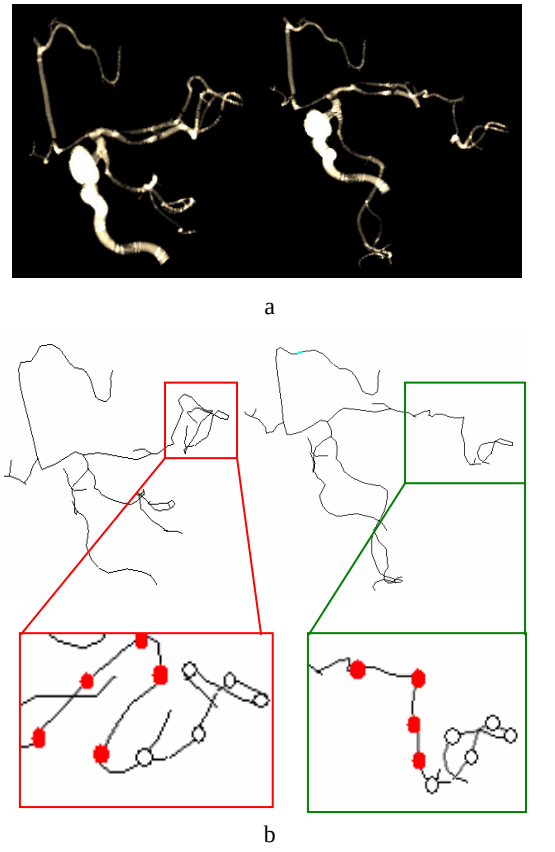


Figure 3. Interactively manipulating branches of a vascular dataset. (a) texture mapped original (left), and manipulated dataset (right); (b) the corresponding IK skeletons are shown below. The area of manipulation is zoomed to shown the exact bone-joint configuration. The red joints have been manipulated.

3.2 Volume Manipulation

An example of a manipulated skeleton for improved visualization is shown in Figure 3. This example utilizes the aneurism dataset. This dataset is very convoluted with many overlapping branches coiled up as shown in Figure 3a(left). The corresponding IK skeleton is shown in Figure 3b(left). A zoom of the skeleton with the joints highlighted is also shown. The red joints are the ones that will be manipulated. In Figure 3a(right), the manipulated dataset is rendered. Figure 3b shows the transformations that were applied to the bones -- basically a “straightening” operation. Once the skeleton is manipulated to the desired pose, the new volume can be rendered and rotated in real time using commodity graphics hardware (see [43] for some movies). The bone manipulation is done through an interactive program called voledit [37]. A picture of the voledit interface is shown in Figure 4. The user can select a bone by clicking on any skeleton segment in the window. Once selected, the bone can be moved interactively. This type of ability is very useful for convoluted shapes, such as the aneurism, where the physician may want to view a particular region in more detail without occluding detail. Allowing the user to interactively examine a dataset, not just on the surface but also the structure can aid in understanding complex shapes.

In the pipeline shown in Figure 2, two different interactive rendering options have been identified (and implemented in

voledit), texture-map based and point-splat based. In Figure 3, texture-mapping was used to render the aneurism. Below we describe both approaches, however, any interactive rendering algorithm can be accommodated by this pipeline (such as, non-photorealistic rendering [30], two-level rendering [19]). Both of these rendering options use the GL library [48].

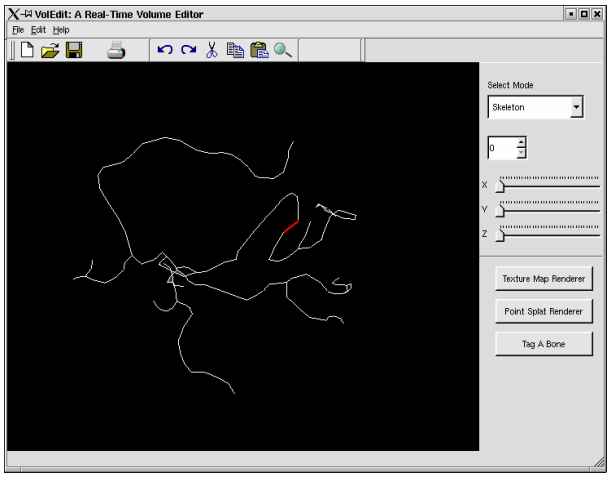


Figure 4. The VolEdit Interface shown with the IK skeleton corresponding to the aneurism dataset.

3.2.1 Texture mapping

A crucial part of enhanced visualization is interactivity. In [13] the reconstruction of the volume is not interactive. In [37] a method was utilized to both manipulate the skeleton and render the volume interactively. The IK skeleton defines a simplified cuboid decomposition of the object that can be used by standard texture mapping hardware [35, 37, 44, 45]. Each cuboid is sliced and texture mapped, and all of the slices are then composited. Since the slicing is viewport-aligned and is done back-to-front, the sectional polygons obtained are sorted along the view direction. When a bone is transformed, a new set of cuboids is computed, and the slice planes are mapped into the original volume (using the graphics hardware) for the appropriate texture. The joints are modeled using mid-plane geometry [2], which involves adding more cross-sectional planes at the joints to prevent cracking and to account for the stretch. This adds a non-linear transformation at the joints. The overall geometry is kept simple to support interactivity with higher frame rates. This method runs at 2-15 *fps* depending on the size of texture, and the underlying geometry. For a lower half of the visible human dataset, a frame rate of 15*fps* was achieved.

3.2.2 Point Splatting

Point splatting can also be used to interactively render a volume. A technique based on elliptical gaussian kernel to render the point splats is presented in [47]. In this work, individual voxels are splatted to the screen by treating each voxel as a vertex, and assigning appropriate texture value based on its spatial coordinates. Each voxel is splatted to the screen based upon the position of its bone. If the bone has been transformed, the voxels associated with that bone also incur the same transformation. The splatted points are then blended in hardware, by utilizing different available compositing options in hardware. The user can interactively choose these options. Voxels must be attached to each bone using the algorithm described in [13, 15, 16]. For large transformations, cracks will occur between joints using this rendering methodology, however, the point size can be increased to provide better coverage (e.g., using `glPointSize`).

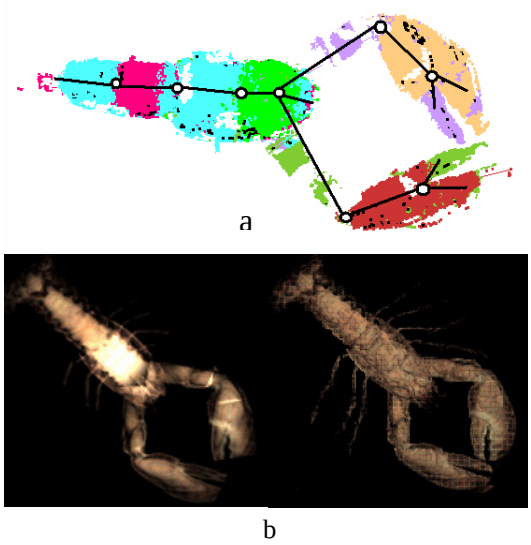


Figure 5. (a) Connectivity of object voxels to IK skeleton. (b) Two modes for rendering: texture map (left), and point splat (right).

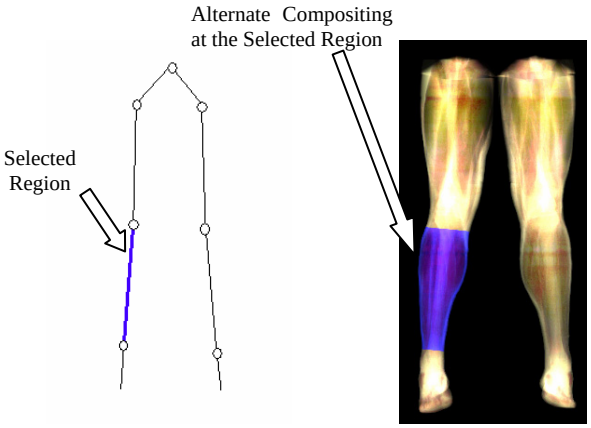
Figure 5a displays the skeleton of the lobster dataset with the skeleton points. The image shows the voxel correspondence to the skeleton (for point splatting), while fitted cuboids are used for texture mapping. In Figure 5b, a comparison of point-splat rendering and texture mapping is shown. Figure 5b(left) shows the lobster rendered using texture mapping and Figure 5b(right) shows the lobster rendered using point splatting. The running time of point-splat rendering algorithm depends directly on the number of skeleton points, and the coloring and compositing are done in hardware. A semi-interative frame rate of 2*fps* was achieved with the 256^3 dataset.

3.3 Selective Rendering

Once a bone is selected, in addition to manipulation, it can also be designated for selective rendering, i.e., just as one can alternately render different segmentation of the volume in different styles [19, 42], one can also render different components of the volume in different style. For example, we may want to render the demarcated component more transparent or more opaque. We may want to use a different transfer function; we may even want to replace that part of the volume with another volume (e.g., if two volumes are available representing the same underlying space). The ability to manipulate/replace logical components with alternate renderings, provides a powerful tool for insightful visualization.

The rendering pipeline can accommodate these changes by loading the new lighting/rendering parameters into the hardware before rendering the components that have been selected. This means that an alternate blending function (`glBlendFunc`), blending equation (`glBlendEquation`), texture environment (`glTexEnv`), or color (`glColor`) can be set for different regions separately. These parameters are applicable to both the rendering mechanisms shown in Figure 2.

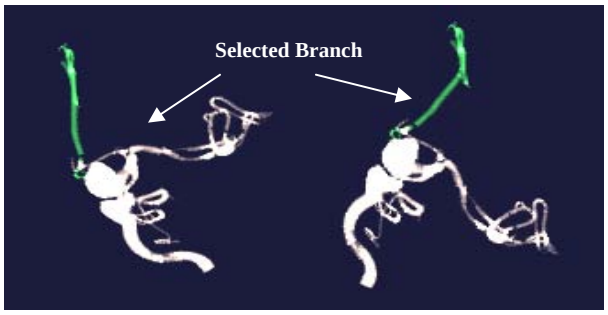
In Figure 6, an example of selective rendering is shown with a different transfer function applied to the selected limb. In this image, the visible human legs are displayed. The left leg has been chosen by the user for selective rendering. In the right image, the leg was rendered with a "bluish" transfer function. Any number of different transfer functions can be chosen for the selected limb. In this case, a simple "coloring" mask was used to highlight that limb.



a b

Figure 6. Selective Rendering: (a) the left shin was selected by the user; (b) alternate compositing is applied to the texture corresponding to the selected region.

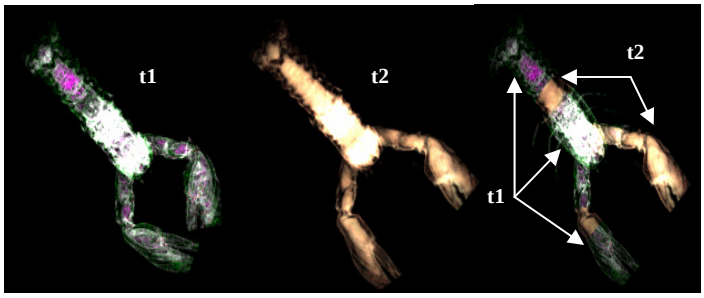
Another example of selective rendering is shown with the aneurism dataset in Figure 7. Since the original dataset is highly convoluted, it is hard to visualize different branches as they are coiled up together. Because of its complexity, the geometry of the dataset is difficult to "see" even during rotation. As previously mentioned, the branches could be uncoiled using the volume manipulation technique. However, once the branches have been moved, it is helpful to demarcate them. By highlighting a branch, the geometry becomes apparent during rotation, and a focus+context view is generated. This is demonstrated in Figure 7. The selected branch has been rendered with an alternate compositing, shown in green in the Figure.



a b

Figure 7. Selective manipulation and rendering. (a) Original aneurism dataset with one branch selected and rendered in green. (b) The branch has been manipulated and rendered to "highlight" it.

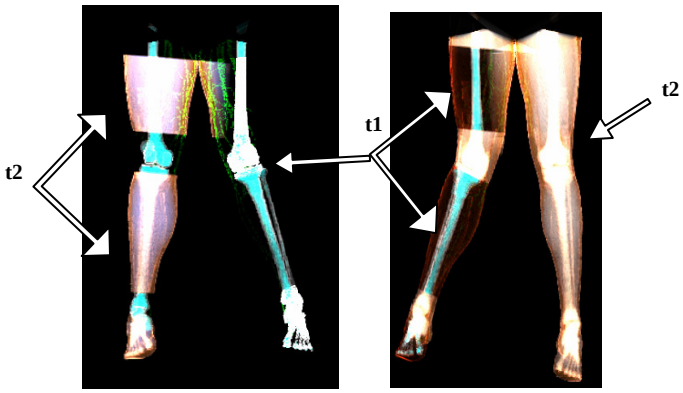
In Figure 8, a lobster is shown with three different renderings applied to selected bones (the skeleton of the lobster is shown in Figure 5). Figure 8a shows the lobster rendered with a transfer function t1 (greenish/purplish color palette); in Figure 8b, the arms have been moved apart and the lobster is rendered with a new transfer function t2 (golden palette); while in Figure 8c, both t1 and t2 have been applied to the dataset in selected regions.



a b c

Figure 8. The lobster dataset. (a) Original lobster with transfer function t1, (b) lobster with moved arms and transfer function t2, (c) t1 and t2 applied selectively to different components of the dataset.

Figure 9 shows an example of selectively determining the transfer functions for different parts of the visible human legs. This dataset is segmented and different tissue types have been previously identified. Two different transfer functions (t1 and t2) have been specified at different positions in the dataset. Note that certain regions have been rendered with more transparency as compared with others. Also, different tissues can be seen, including the nerves in green (Fig. 9a), bones, and skin.



a b

Figure 9. Selective transfer function determination for parts of the visible human legs. Two transfer functions t1, and t2 have been used. In the left image, t1 and t2 have been attached to different portions of the legs; in the right image, t1 and t2 positions have been interchanged. t1 colors the nerves and bones, whereas t2 shows all the tissues.

Another example of specifying transfer function at logical components of a volume is illustrated in Figure 10. The Figure shows the upper portion of the visible human dataset. Two transfer functions (t1, t2) have been used in this example. In Figure 10a, the head and the left arm have been rendered with transfer function t1, while the rest of the torso uses transfer function t2. In Figure 11b, t1 has been specified at the central torso region, while in Figure 11c it has been extended to the shoulders. The images are rendered using the volumetric point splatting technique (discussed in Section 3.2.2).

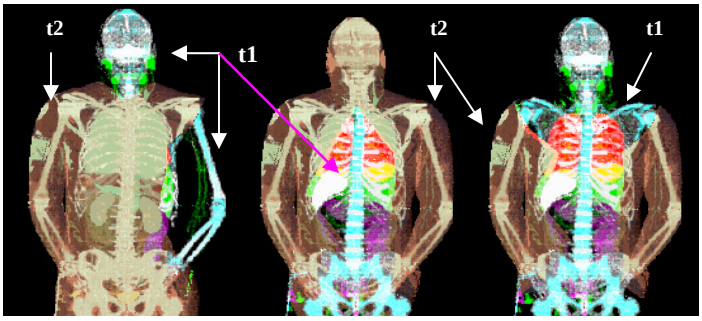


Figure 10. Point-splat rendered upper torso of the visible human dataset. Two transfer functions (t1, t2) have been used in this example. From a to c, the two functions have been specified at different regions in varying proportion.

The Bonsai dataset, is shown in Figure 11. This dataset has a large number of branches. One intuitive manipulation could be to move the leaves/branches to see the stems underneath. In the figure, the stems have been selectively rendered with a "brownish" transfer function, and the leaves with a "greenish" transfer function. Figure 11a shows the original dataset, and Figure 11b shows the manipulated branches. The leaves were manipulated towards the right to show the trunk. Also, the trunk was moved to stand upright.

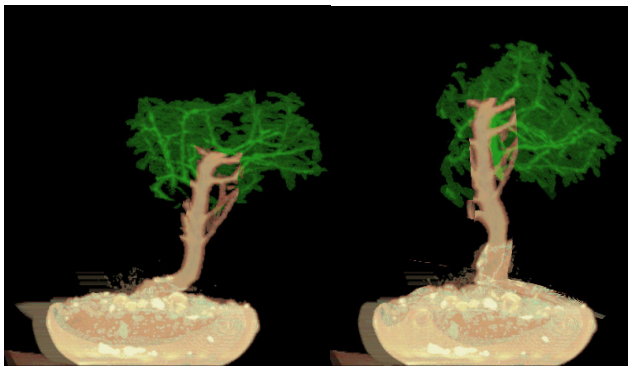


Figure 11. The Bonsai dataset. (a) Original dataset with leaves colored with a greenish transfer function and the stems with a brownish transfer function, (b) leaves moved to the right to expose the stem on left (also the trunk has been moved to stand straight).

3.3.1 Selective Compression / Selective High-Resolution

Another aspect of selective rendering is to apply higher resolution to a selected region (*selective de-compression*). This feature assigns higher resolution to a portion of the volume while still embedding it in the context of the entire volume, thereby creating a literal focus+context view. In Figure 12, we show the upper torso of the visible human dataset. The left arm was selected, and was rendered using a higher resolution. The higher resolution is obtained from another dataset, since the information about logical component is the same at all resolutions (e.g., the right arm is the right arm in all the resolutions). The higher resolution arm has better nerve resolution as shown in Figure 12b. In order to apply higher resolution at a selected portion, the voxels from the higher resolution were scaled to the appropriate location in the model. In Figure 12, point-splat rendering is used. For texture mapped rendering, the portion of the higher resolution dataset that is needed must be loaded into the texture memory.

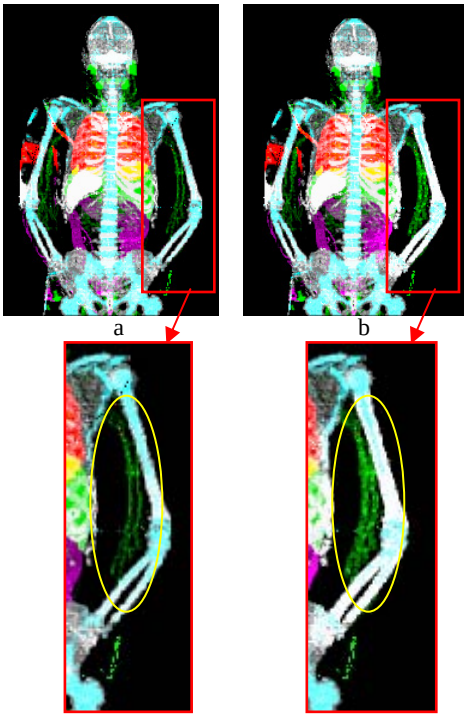


Figure 12. Selective compression of the visible man arm. In (a), the low resolution visible man is used (`196x114x626`). In (b) the arm was selected for higher resolution and the arm from a `586x340x1878` dataset is substituted.

3.3.2 Selective Juxtaposition

The selected components can be replaced with data from other datasets (selective juxtaposition). This feature enables the user to place parts from two or more datasets side-by-side together into the same space. The user selects the regions to be juxtaposed with other datasets from the IK skeleton, and associates each of them to correspond to the datasets. During the rendering pass, the system recognizes the tagged ("selected") bones and the corresponding datasets, and renders each region separately. The datasets need not be simultaneously loaded into the hardware texture memory. This allows for better utilization of the graphics hardware resources.

In Figure 13a, the legs of the visible human dataset are shown, however, they belong to different datasets. The two datasets are different with respect to the tissue segmentation. The right leg corresponds to one version of the visible human dataset, while the left one corresponds to another version. The same transfer function was applied to both the legs. In Figure 13b, the two legs were rendered with different transfer functions. Note the appearance of nerves and bones in the left leg.

Another example of selective juxtaposition is shown in Figure 14. In this figure, the lobster arm was mapped to the human arm. As before, the logical components were obtained from two different datasets. In Figure 14a, the visible human arm in red was chosen by the user to be replaced with the lobster's arm. The bone corresponding to the lobster's arm is transformed to map to the human arm. This type of juxtaposition may involve a non-linear transformation if it requires to scale the two objects for spatial

mapping. Figure 14b(right) shows a straight-down manipulation applied to the replaced lobster arm. For texture mapping, since both use bounding cuboids, the cuboids have to be mapped to one another. For point-splatting, this reduces to mapping the bone segments. Both of these processes are similar to morphing [20, 21]. More examples can be seen on [43].

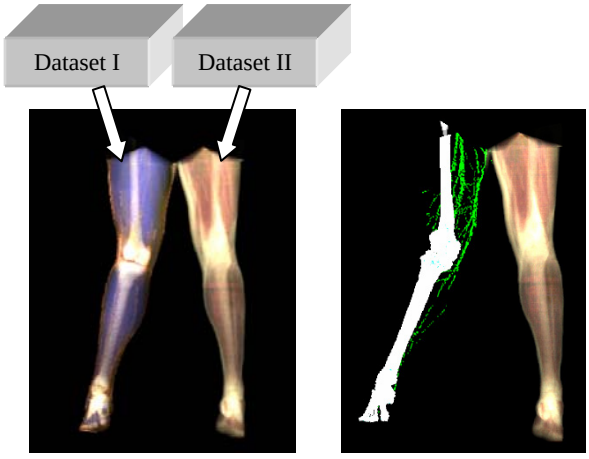


Figure 13. Selective juxtaposition. In each image, right leg belongs to the original dataset, and left to a different dataset: (a) same transfer function applied to both legs; (b) different transfer function applied to both legs. Note the nerves (green) and bones (white) in the leg corresponding to the second dataset.

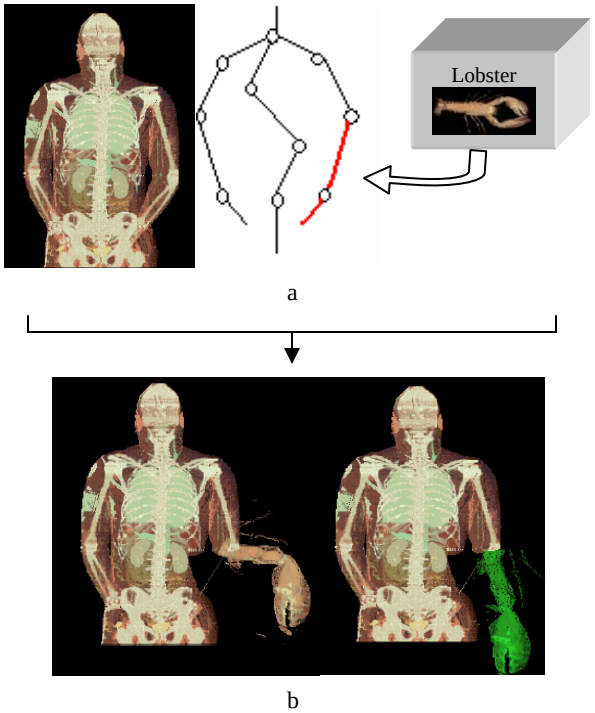


Figure 14. Selective juxtaposition of the visible human arm with lobster's arm. (a) the arm shown in red was picked by the user to replace with lobster's arm; (b) two poses are shown to illustrate the juxtaposition with manipulation. The left image shows the juxtaposed arm. In the right image, the arm was manipulated down and rendered with an alternate "greenish" transfer function.

The application was implemented on a Linux based PC, with 1.7GHz P4 processor and 64MB Geforce3 GPU. The system operates at about 2-15fps, depending upon the size of the datasets (discussed in Section 3.2.1 and 3.2.2) and the type of rendering used. The skeletonization is done as a preprocessing step. Currently, the interface (shown in Figure 4) allows a user to select a bone(s) and assign an alternate color map or rendering option to it. We are improving this interface to allow any greater flexibility in determining the rendering type (e.g. shading, lighting model, etc.).

4 CONCLUSIONS

We have presented a methodology to facilitate improved visualization of volumetric datasets. The methodology, which is skeleton based, supports both manipulation and selective rendering. The skeleton decomposes the volume into a set of connected components (graph) creating a more intuitive method to select regions of interest for manipulation or display. Manipulation allows the user to "move" occluding regions or uncover twisted geometries. Selective rendering allows the user to interactively generate focus+context style of views by selecting regions of the data to "highlight". New transfer functions can be applied to the selected regions, higher resolution data can be rendered, or regions from different data substituted. In this paper, we have shown examples of all of these types of renderings applied to different volumetric datasets.

5 ACKNOWLEDGEMENTS

This research at Rutgers University's Vizlab is supported by the National Science Foundation under grant number NSF 0118760.

REFERENCES

[1] Character Studio Animation Toolkit http://www.discreet.com/products/cs/

[2] J. Chadwick, D. Haumann, R. Parent. Layered Construction for Deformable Animated Characters. In *Proceedings of SIGGRAPH 89*, 23(3), pages 243-252, July 1989, Boston, Mass.

[3] M. Chen and J. Tucker, Constructive Volume Geometry, Computer Graphics Forum, 19(4), 2000.

[4] M. Chen, D. Silver, A. Winter, V. Singh, and N. Cornea, Spatial Transfer Functions – A Unified Approach to Specifying Deformations in Volume Modeling and Animation, V*olume Graphics 03, July2003.*

[5] Y. Chen, Q. Zhu and A. Kaufmann. Physically-based Animation of Volumetric Objects. In *Proceedings of IEEE Computer Animation '98*, pages 154-160, 1998.

[6] C. Chua, and U. Neumann. Hardware-Accelerated Free-Form Deformations. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2000.

[7] B. Csebfalvi, L. Mroz, H. Hauser, A. kong, and M. Groller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3): 452-460, September 2001.

[8] H. Doleisch, H. Hauser, Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. Technical Report, VRVis, Austria.

[9] D. Ebert, P. Rheingans, Volume Illustration: Non-photorealistic Rendering of Volume Models, In *Proceedings of IEEE Visualization*, 2000.

[10] K. Engel, M. Kraus, T. Ertl, High-quality Pre-Integrated Volume Rendering using Hardware-Accelerated Pixel Shading, *SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, 2001.

[11] S. Fang and R. Srinivasan. Volumetric CSG – A Model-Based Volume Visualization Approach. In *Proceedings of 6th International Conference in Central Europe on Computer Graphics and Visualization*, pages 88-95, 1998.

[12] G. Furnas, The Generalized Fisheye Views. In *Proceedings of CHI '86*, pp. 16-23, ACM Press, 1986.

[13] N. Gagvani and D. Silver. Animating Volumetric Models. *Graphical Models and Image Processing, Academic Press*, March 2002.

[14] N. Gagvani and D. Silver, Parameter Controlled Volume Thinning. *Graphical Models and Image Processing, Academic Press*. Vol. 61, no. 3, 1999.

[15] N. Gagvani, and D. Silver. Animating the Visible Human Dataset. *Proceedings for The Visible Human Project Conference*, 2000.

[16] N. Gagvani, and D. Silver, Realistic Volume Animation with Alias, *Proceedings of International Workshop on Volume Graphics*, March 1999.

[17] N. Gagvani, D. Hosekote, and D. Silver, Volume Animation using the Skeleton Tree, In *Proceedings ACM/IEEE Symposium on Volume Visualization '98, RTP, North Carolina, pp. 47-53*, 1998.

[18] S.F. Gibson. 3D Chain Mail: A Fast Algorithm for Deforming Volumetric Objects. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, April 1997.

[19] M. Hadwiger, C. Berger, H. Hauser, High-Quality Two-Level Volume Rendering of Segmented Data Sets on Consumer Graphics Hardware, In *Proceeding of IEEE Visualization*, 2003.

[20] T. He, S. Wang, and A. Kaufmann. Wavelet-Based Volume Morphing. In *Proceedings of Visualization 94*, pages 85-91, Los Alamitos, CA, 1994, IEEE Computer Society Press.

[21] J. Hughes, Scheduler Fourier Volume Morphing, *Computer*, 26, 2(July 1992), 43-46.

[22] The Institute of Electrical and Electronics Engineers Inc., http://www.ieee.org

[23] S. Islam, S. Dipankar, D. Silver, M. Chen, Spatial and Temporal Splitting of Scalar Fields in Volume Graphics, *IEEE Volume Visualization*, 2004..

[24] O. Kreylos, N. Max, B. Hamann, S. Crivelli, and E. Bethel, Interactive Protein Manipulation. *IEEE Visualization 03*, 2003.

[25] Y. Kurzion and R. Yagel. Space Deformation using Ray Deflectors. *6th EuroGraphics Workshop on Rendering 95*, pages 21-32, 1995.

[26] Y. Kurzion, and R. Yagel. Interactive Space Deformation with Hardware Assisted Rendering. *IEEE Comp. Graphics and Apps.*, 17(5), 1997.

[27] S. Lakare, A. Kaufman, OpengVL, The Open Volume Library, In *Proceedings of International Workshop on Volume Graphics*, Japan, 2003.

[28] J. Lamping, R. Rao, and P. Pirolli, A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *CHI Proceedings*, 1995.

[29] J. Lamping, R. Rao, P. Pirolli, A focus+context technique based on hybolic geometry for visualizing large hierarchies. In *Proceedings of CHI'95*, pp. 401-408, ACM Press, 1995.

[30] A. Lu, C. Morris, D. Ebert, P. Rheingans, C. Hansen, Non-photorealistic Volume Rendering using Stippling Techniques, In *Proceedings of IEEE Visualization 2002*, 2002, Boston-Massachusetts.

[31] M.J. McGuffin, L. Tancau, R. Balakrishnan. Using Deformations for Browsing Volumetric Data. In *Proceedings of IEEE Visualization*, 2003, pp 401-408.

[32] Maya: A 3D Animation and Visual Effects Tool http://www.aliaswavefront.com/en/products/maya/index.shtml

[33] National Library of Medicine, http://www.nlm.nih.gov/

[34] I. Palmer, and R. Grimsdale, Collision Detection for Animation Using Sphere-Trees, *Computer Graphics Forum*, 14(2):105-116, 1995.

[35] C. Salama, M. Scheuering, G. Soza, and G. Greiner, Fast Volumetric Deformation on General Purpose Hardware. In *Proceedings of SIGGRAPH/EUROGRAPHICS Graphics Hardware Workshop 2001*, pages 17-24, 2001.

[36] D. Silver, and N. Gagvani, Unwinding the Colon, *Medicine Meets Virtual Reality (MMVR)*, 2002.

[37] V. Singh, D. Silver, and N. Cornea, Real-Time Volume Manipulation, *Volume Graphics 03, July 2003.*

[38] R. Spence, M. Apperley, Database Navigation: An Official Environment for the Professional. *Behavior and Information Technology*, Vol. 1, No. 1, pp. 43-54, 1982.

[39] M. Tory, T. Möller, M. Atkins, and A. Kirkpatrick, "Visualization Task Performance with 2D, 3D, and Combination Displays", *IEEE Transactions on Visualization and Computer Graphics*, 2003.

[40] The VolVis website. http://www.volvis.org/

[41] vtk Toolkit, Kitware Inc. http://www.kitware.com

[42] vlib – A Volume Graphics API, http://vg.swan.ac.uk/vlib/

[43] VIZLAB, Rutgers, The State University of New Jersey, http://www.caip.rutgers.edu/vizlab.html

[44] R. Westermann and T. Ertl. Efficiently using Graphics Hardware in Volume Rendering Applications. In *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 291-294, 1998.

[45] R. Westermann and C. Salama. Real-Time Volume Deformation. In *Computer Graphics Forum (EuroGraphics)*. 2001.

[46] X. Yuan, D. Silver and B. Raman, Computing the Curve-Skeletons of General 3D Objects, *Submitted for publication.*.

[47] M. Zwicker, H. Pfister, J. Baar, M. Gross, EWA Volume Splatting, *IEEE Visualization*, 2001.

[48] The OpenGL Website, http://www.opengl.org