Eurographics Workshop on Visual Computing for Biomedicine (2008)
C. P. Botha, G. Kindlmann, W. J. Niessen, and B. Preim (Editors)

Focus+Context rendering of structured biomedical data

P. Abellán [1] and A. Puig [2] and D. Tost [1]

[1]CG- Research Center in Biomedical Engineering, UPC, Barcelona, Spain
[2]Dep. Applied Mathematics, UB, Barcelona, Spain

Abstract

Biomedical data can be classified according to different taxonomies. Understanding the relationships between different data categories is essential for an in-depth knowledge of the data. We present a volume rendering system aimed at outlining structural relationships between different classification criteria of a biomedical voxel model. The system clusterizes the model into subsets of voxels sharing the same classification criteria. It constructs a labelled voxel model storing for each voxel an identifier of its associated cluster. We represent the classification space as a graph and we render it in the application interface. This way, clinicians can specify their visualization queries by selecting nodes of the graph and boolean operations between them. Given a rendering query, the system computes a transfer function on the labelled voxel model domain. This transfer function, together with the original voxel model and the labelled voxel model, are used during rendering to visualize the selected data more or less colored according to level of the graph at which they have been selected, and contextualized with the other parts of the model to which they are related. We demonstrate the utility of our approach on several biomedical datasets.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Three-dimensional graphics and realism

1. Introduction

Today, a major concern in biomedical visualization is to provide meaningful images conveying relevant information on the data. As datasets are each time larger, they contain more information than users can cope all at once. Moreover, precisely because of this data complexity, clinicians have more difficulties to isolate relevant parts of the data with conventional interfaces. This is why, inspired in classical illustration techniques, new means of outlining significant features are being developed, such as cut-aways, ghost views, silhouette enhancement, Focus+Context (F+C) and adaptive shading.

Biomedical data can be organized into taxonomies according to very diverse criteria such as functional system, tissue type, anatomical structure and activity level. A good understanding of the relationships between these different criteria is essential in the acquisition of an in-depth knowledge of the information conveyed in the images. Current rendering systems can outline regions of interest, but they do not show the structural relationships existing in the data. Moreover, structures as the human brain are composed of a large number of parts, some of them tiny and difficult to select on a 3D rendered image. The system that we describe in this paper provides means of specifying regions of interest as a combination of classification criteria. Clinicians can use it to select parts of the biomedical model using a graphical representation of the structure of the dataset labelled with the medical names of the structures. The system renders the selected regions isolatedly or put them into context with other regions having a structural relationship with them. We call this type of rendering *Structural Focus and Context* (SF+C).

The paper brings four contributions: first, the modeling of the relationships between isoclassified subsets of biomedical data; next, the computation of the visibility and color of these subsets according to SF+C rules; third, the visualization of the data through a 3D texture mapping rendering system; and, fourth, an intuitive widget for the interactive selection of the rendered features and their structural context.

2. Previous work

The previous work related to ours falls into three categories: F+C techniques, volume structuring and query-driven visualization.

The idea of visually emphasizing certain structures in a dataset is based on the observation that cognitive understanding is easier when the observer's attention is concentrated on few stimuli. Driving users attention on relevant features can be accomplished by using several complementary approaches: camera adjustments, clipping, ghosting and multiple shading styles. Setting the camera so that it focuses on the relevant structures of the model requires sometimes a lot of user expertise. Therefore, many authors [VFSG06] have addressed viewpoint optimization and distortion lenses [WZMK05]. Since relevant structures of the volume can be occluded by others, Bruckner et al. [BE06] propose to generate exploded views of a volume. Alternatively, Wang et al. [WK95] propose to explore volumes using sculpting clipping tools. Weiskopf et al. [WEE02] apply cut-away in 3D-texture mapping rendering. Owada et al. [ONOT04] describe a system that shows textured arbitrary cuts of surface models. Finally, Correa et al. [CCS06] generalize the concept of virtual cuts and deformation by defining feature-aligned manipulation operators. Instead of clipping non-relevant structures, many authors propose to show them but faded out with high transparency, low resolution and different shading style, so that they only contextualize the region-of-interest. The semantic depth-of-field uses selective blur to make less important objects less prominent [KMH*02]. Depth-peeling has been used to make more transparent peripheral regions of the volume [RSK06]. Ghosting represents less-important regions with high transparency and relevant features with higher opacity [BGKG06]. Context surfaces can be outlined by modulating the opacity according to the gradient value [Lev91] and by properly designing the transfer function [LM04]. Sparse representation of context is achieved by rendering only the contour of these structures [CMH*01]. Viola et al. [VKG05] propose several techniques to prevent an object from being occluded by another less important one, as for instance the screen door transparency, a technique in which the occluder is painted as a wire mesh with holes in it. Moreover, different shading styles can be used for focus and context [LME02]. Rautek et al. [RBG07] introduce the concept of semantic layer that define the mapping of volumetric attributes to one visual style. Features can also be rendered separately with different algorithms and the resulting images can combined [HMBG01] [VKG05]. The ClearView system [KSW06] renders the context by layers, each layer separately in a different texture. Then, it composes the textures according to the 2D user-defined position of the focus and applying importance-based shaders based on the curvature, the relative distance and the view distance of the texels. The medical illustration system designed by Svakhine et al. [SES05] integrates boundary and silhouette enhancement with different shading styles, including toon.

The systems described so far provide valuable illustrative images but they do not focus on the visualization of the structure of biomedical data. However, even apparently simple biomedical datasets can have complex structural relationships. The analysis of these data can benefit from an explicit modeling of their structure. Car et al. [ACS03] represent the hierarchical structure of iso-valued surfaces within a volume dataset as a *contour tree*. Weber et al. [WDC*07] use different transfer functions to render pre-segmented regions of equivalent contour topology. Nevertheless, these approaches are restricted to value-based classification criteria and hierarchies. Nadeau [Nad00] defines the *scene graph* to express a multi-model scene as a graph of volume objects and filter operations. This is a constructive geometry approach in which the graph reflects users manipulations rather than intrinsic data structuring. None of these approaches apply F+C techniques.

Expressing visualization queries as boolean compound of values or color ranges can be computationally expensive if the expression is evaluated for every data sample and the dataset is very large. Query-based visualization addresses this problem. In particular, Stockinger et al. [SSBW05] design a bitmap indexing scheme (DEX) that efficiently answers multivariate and multidimensional queries. Ferré et al. [FPT06] propose to run-length codify combinations of data values in order to skip unselected regions during data traversal. Bruckner et al. [BG05] codify the selected volume with an auxiliary volume model that is modified on user query. They define three types of regions: the selection, the ghost and the background. Burns et al. [BHW*07] address illustrative visualization of an object of interest embedded into context volume data. They define a flexible cut-away importance-driven structure.

3. Overview

Figures 1 and 2 illustrate the pipeline of our method. First, in a pre-process, the initial voxel model V is classified according to various independent criteria. In our simulations we have used hand-labeled anatomical regions, value-based classifications and spatial partitioning. Moreover, in MR brain models we have used an additional registered SPECT dataset as a classification criteria of the activity level. In the second step of the pre-process we clusterize the voxels of the model according to the combination of classification criteria that they fulfill. We construct a labelled voxel model LV that stores, for each voxel, a unique identifier of the cluster to which this voxel belongs. We represent the classification space as a directed graph G such that its nodes represent classification criteria and its edges represent logical implications between criteria. Each leaf node of the graph is associated to one or more clusters.

Figure 3 shows a very simple illustrative example. We have classified a CT foot dataset according to three different criteria (foot and not_foot; ankle, palm and toe; toe_1 to toe_5 and not_toe). After removing the empty regions, the constructed graph, shown in the middle of the image, has 10 nodes. The edges between nodes indicate that a classification criterion implies another: for instance, being *toe_1* implies being *toe*, and being *toe* implies being part of *foot*.

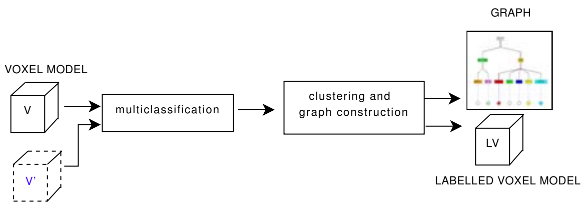


Figure 1: *Pre-process of the proposed method. The voxel model V is classified according to various criteria, eventually taking into account other data modalities V'. The graph and the labelled voxel model LV are constructed.*

The combination of classification criteria defines seven clusters (c1 to c7) shown at the bottom part of the graph. Cluster c1, for instance is the set of voxels that fulfill the criteria of being part of the foot, not being a toe and being part of the ankle. At the left side of the image, each cluster of the labelled voxel model is shown with a different color.

The rendering stage is divided into two steps. In the first stage, users interactively express their rendering query as a boolean expression of classification criteria. Given the query, a parser traverses the graph in order to identify the selected clusters. The parser computes a transfer function LTF in the labelled voxel model LV domain that denotes color, opacity and gradient modulation factor of the different clusters. The second step takes as input the original voxel model V, its original transfer function TF, the labelled voxel model LV and its computed transfer function LTF. Rendering is performed by applying 3D texture-mapping. The fragment shader fetches the voxel value in V and the cluster identifier in LV and, from them, the corresponding rendering parameters in TF and LTF, respectively. Shading merges the rendering parameters according to a user-specified weight.

The right-most image of Figure 3 shows the rendering of the example query to draw *palm* contextualized with the other nodes of the graph to which it is related. Since the parent node of *palm* is node *foot*, the computed transfer function LTF sets to very low opacity values and white color all the clusters but c1. The color and opacity of cluster c1 are a mixture of its color and opacity in TF and LTF.

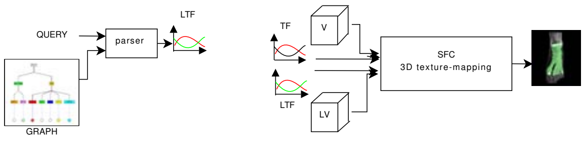


Figure 2: *Rendering step. Left: first step, given a user query and the graph, the transfer function LTF is computed. Right: second step, 3D texture mapping handling simultaneously the original voxel model V and the labelled voxel model LV together with their transfer functions TF and LTF.*

With this strategy, we are able to enhance focus features and to show them in relation to the other structures of the graph. The contextual information is structural, this is why we define this type of visualization as *Structural Focus+Context*. We are able to visualize the structures that include the ones into focus at different levels of depth and to show other structures sharing a common ancestor. Moreover, the widget that we have designed for the specification of the focus features, provides a visual feedback of the colors and opacity with which all the structures of the model are rendered. This gives to users interesting clues to understand the structural semantics of the image.

4. Multiclassified set structure model

4.1. The structure graph *G*

Let V be the voxel model and nc be the number of independent classification criteria defined on V. Each classification criterion c_i, $i = 1 \dots nc$ separates the voxel model into a finite set of nc_i disjoint classes. We denote as c^i_j the jth class identifier of the ith criterion. We call *isoclass(c^i_j)* the set of voxels of V that share the classification identifier c^i_j. Since isoclasses of the same criterion are disjoint, it is fulfilled that:

- $\forall j,k : 1 \leq j,k \leq nc_i, j \neq k : isoclas(c^i_j) \cap isoclas(c^i_k) = \emptyset$
- $\bigcup_{j=1}^{nc_i} isoclas(c^i_j) = V$

On the contrary, isoclasses of different criteria are not necessarily disjoint. Even more, one isoclass can include another one. Specifically, we say that a class identifier c^i_k implies a class identifier c^j_l if the isoclass of c^j_l includes all the voxels of the isoclass of c^i_k, being $i \neq j$:

$$c^i_k \Rightarrow c^j_l \texttt{ iff } isoclas(c^i_k) \subset isoclas(c^j_l)$$

We denote as $I = \{c^j_i, 1 \leq i \leq nc, 1 \leq j \leq nc_i\}$ the set of all isoclass identifiers. We call E the set of all implication relationships between isoclass identifiers. Therefore, we represent the structure of V as a directed graph $G = (I, E)$ such that the nodes represent isoclass identifiers and the edges implications between nodes. Edges go from inclusor isoclasses to included ones. Since the inclusion relationship is antisymmetric, G is acyclic. Thus, it has leaf nodes from which no edges exit. It may have many roots, and it can be a tree only in very simple cases as the one shown in Figure 3.

4.2. The labeled voxel model *LV*

For each voxel v of the voxel model V we can compute a nc_tuple of class identifiers, one for each criterion. During the process of graph construction, we separate the voxels into *clusters*, i.e. sets of voxels sharing the same isoclass identifiers tuple. There are potentially as many clusters as combinations of isoclasses, however, we consider only the non-empty sets. Therefore, the number of clusters nr is such that $1 \leq nr \leq \prod_{i=1}^{nc} nc_i$. We denote R the set of all the clusters: $R = \{r_i, i \in \{1 \dots nr\}\}$. Again, it is fulfilled that all the clusters are disjoints and their union is the voxel model.

We define the labelled voxel model LV with the same resolution and spatial orientation as V and such that the value of every voxel in LV is the identifier of the cluster to which the voxel belongs. The labelled voxel model LV is used as a selection mask of V. The correspondence between the leaf nodes of the graph and the voxels of LV is established through a binary relation $\mathcal{F}$ with domain the isoclass identifiers set I and with codomain the clusters set R. This relation associates to a class identifier $c_i^j \in I$ a cluster $r \in R$, if c_i^j has no descendant in graph G and if all voxels of r fulfill the classification criterion c_i^j.

4.3. User queries and parser rules

In order to explore the structure of the dataset, users define regions of focus as union, intersections and complementary sets of the isoclasses. We express user queries as boolean expressions defining the isoclasses to which the selected voxels belong. Specifically, let $b_k^i(v)$ be a boolean expression that indicates if voxel v ($v \in V$) belongs to the isoclass identified by c_k^i: $b_k^i(v) \Leftrightarrow v \in isoclass(c_k^i)$. A user query can be expressed as a combination of b_i^j with the operators $\wedge, \vee$ and $\neg$. These combinations define unequivocally a subset of V that can be expressed in terms of unions of clusters of LV. We call $\mathcal{M}$ the function that associates to any user query a unique union of clusters in R. It is based on the following rules: let C_r be the nc_tuple of class identifier of a cluster r, for all $i, j \in \{1 \dots nc\}$, $k, k1, k2 \in \{1 \dots nc_i\}$ being $k1 \neq k2$, and $l \in \{1 \dots nc_j\}$:

- $\mathcal{M}(b_k^i) = \bigcup_{m=1}^{nr} r_m$ such that $c_k^i \in C_{r_m}$
- $\mathcal{M}(\neg(b_k^i)) = \bigcup_{m=1}^{nr} r_m$ such that $c_k^i \notin C_{r_m}$
- $\mathcal{M}(b_k^i \cup b_l^j) = \bigcup_{m=1}^{nr} r_m$ such that $c_k^i \in C_{r_m} \vee c_l^j \in C_{r_m}$
- $\mathcal{M}(b_{k1}^i \cap b_{k2}^i) = \emptyset$
- $\mathcal{M}(b_k^i \cap b_l^j) = \bigcup_{m=1}^{nr} r_m$ such that $c_k^i \in C_{r_m} \wedge c_l^j \in C_{r_m}$

The parser implements these rules. It traverses the graph until reaching nodes that are $\mathcal{F}$-related to clusters. Throughout the traversal, it computes the list of bounding boxes of the selected clusters and the transfer function LTF following the coloring rules described in Section 6.

5. Graph Construction

The graph construction proceeds in two steps. First, we construct the isoclasses by separating each classified voxel model into sets. At this stage of the process, we also compute the number of voxels of the isoclasses and their bounding box. We set the opacity of the isoclass as a factor of their bounding sphere radius and their number of voxels. These are the basic opacities of the isoclasses that are combined for rendering according to the algorithm exposed in Section 6.2 and taking into account F+C rules. We automatically compute nodes colors according to Stone's coloring model [Sto06], but we also let users interactively editing them at their convenience. We create one graph node per isoclass. Then, we compute for each isoclass the set of isoclasses that it contains. For that, we construct an adjacency table that stores the number of voxels of the included isoclasses and a zero value for the non-included isoclasses. We sort the raws of the adjacency table by increasing number of voxels, in order to be able to derive the inclusion hierarchy between nodes through a simple matrix traversal. We perform this process on a per-slice basis, in order to avoid an excessive memory load for huge and complex models.

In the second step of the process, we construct the labelled voxel model LV and the list of clusters. For each voxel v, we identify the set of isoclasses to which the voxel belongs. We compute the identifier of the corresponding cluster as a hash code of the isoclasse identifiers. We label the voxel with this identifier in LV and, if the cluster identifier is new, we insert it in the clusters list. The relationship $\mathcal{F}$ between the graph nodes and the clusters is codified in the hash code.

6. Rendering

The rendering step is based on 3D texture-mapping. The fragment shader fetches the property value p in the original voxel model V and the cluster identifier r in LV. Since both models are aligned, a unique geometrical transformation per pixel is needed. Tri-linear interpolation is set in V and nearest-neighbor in LV. The cluster identifier r is used to fetch the rendering parameters of LV in the transfer function LTF. The property value r is used to fetch the voxel color and opacity in V's transfer function TF. The fragment shader combines these values in the shading equation, getting structural cut-away, ghosting and coloring effects. After texture slicing, on user query, all the bounding boxes computed by the parser can be rendered in wire frame.

For very large datasets such as the Visible Male (see Section 8), that do not fit into one texture, during the graph construction, we subdivide V into a set of disjoint, axis aligned bricks. We store the set of bricks to which each cluster belongs. The graph traversal returns the list of selected bricks bounding boxes. Since the boxes are axis aligned and disjoint, we trivially sort them according to the current viewpoint. The rendering step processes this list orderly and, for each brick, it loads the texture before rendering it.

6.1. Structural coloring

The graph provides a natural way to show with colors the structure of classes to which a voxel belongs. Each classification criterion independently identifies each of its classes with a different color. The color associated to each class is stored as an attribute of the corresponding node in the graph. Given a user query, during the graph traversal, the parser computes the clusters colors in the LTF transfer function. In the rendering step, the clusters colors are blended with the voxels colors according to a user-specified blending factor β, $0 \leq \beta \leq 1$.

We have tested several color arithmetics for the computation of LTF. The results shown in Section 8 are based on the following color arithmetics: let $c1 = (r1, g1, b1)$ and $c2 = (r2, g2, b2)$,

- $\overline{(c1)} = (MAX - r1, MAX - g1, MAX - b1)$, being MAX the maximum intensity
- $c1 \cup c2 = (max(r1, r2), max(g1, g2), max(b1, b2))$
- $c1 \cap c2 = (min(r1, r2), min(g1, g2), min(b1, b2))$

The parser applies the following rules. Let $col(b^i_j)$ be the color of isoclass b^i_j, for $i \in \{1 \ldots nc\}$ and $j \in \{1 \ldots nc_i\}$,

- if $r = \mathcal{M}(\neg b^i_k)$ then $LTF(r) = \overline{col(b^i_k)}$
- if $r \in \mathcal{M}(b^i_k \cup b^j_l)$ then $LTF(r) = col(b^i_k) \cup col(b^j_l)$
- if $r \in \mathcal{M}(b^i_k \cap b^j_l)$ then $LTF(r) = col(b^i_k) \cap col(b^j_l)$

6.2. Structural Cut-away and structural ghosting

Cut-away is achieved through the definition of the α-channel of the transfer function (LTF). The combination of classification criteria expressed by users define the union of clusters on focus. In order to cut-away the other clusters, the parser sets the opacity channel with binary values, 1 to the selected clusters and 0 otherwise. Thus, the shader does not render fragments with zero opacity value.

In order to contextualize the focus into the structure to which it belongs, users can define the *ghosting level*. This is an integer value ranging between 0 (cut-away) to the maximum depth of the hierarchy of trees inside the graph. Specifically, a ghosting level of 1 shows as context all the clusters that have a direct common ancestor with the focused clusters. A ghosting level of 2 shows the clusters having common ancestors with the focused clusters through a one-edge path in the graph, and so on with higher ghosting levels.

The parser computes non-zero opacity values for the contextual clusters in the α-channel of LTF. This computation is done after the user query has been evaluated and the focus clusters have been determined. Starting from the graph nodes $\mathcal{F}$-related to those clusters, it traverses the graph up to the *ghosting depth*. It computes the opacity of the contextual clusters according to a factor inversely proportional to the depth of the nodes that are related to them. Therefore, the further in the hierarchy is a node, the lighter is the opacity of the related cluster.

The color of the contextual cluster can be computed similarly as that of the focus, applying structural coloring (see Section 6.1). Alternatively, if a *color_ghosting* flag is activated, the parser can set the values of LTF to a gray-scale for the contextual clusters, taking into account only the nodes intensity rather than their RGB values. The use of gray scale for the context enhances by contrast the perception of the colored focus.

6.3. Shading

The shader computes surface shading (Phong), emission and absorption or both models for the focus data as well as the context data. The shading type is a global parameter of the application. Users define the type of visualization of the context: all the context voxels, only boundary context voxels or only bounding boxes. In the second case, we modulate the width of the context surfaces according to the gradient modulation computed by the parser and stored in LTF. This factor is related to the *ghosting depth* so that, the deepest in the hierarchy is a context cluster, the thinner and most transparent its surface. When the context bounding boxes are shown, the parser sets to zero the opacity of LTF in order to prevent context voxels from being rendered.

6.4. Camera and clipping

The view reference point of the camera is computed as the center of the clusters on focus. However, the projection window is computed as the bounding box of the focus and the contextual clusters. This way, the whole contextual information is visible but the images are centered on the focus. Moreover, during 3D texture rendering, we apply a per-plane opacity modulation to all the proxy polygons that are in front of the focus bounding box. This makes more transparent the slices that traverse only contextual information. Currently, we do not apply view point optimization, therefore the camera orientation is specified interactively by users.

The interface allows users interactively defining clipping planes. We set the initial position of these clipping planes at the bounding box of the focus. Moreover, our application is able to load a third voxel model that contains the distance map of all the voxels to the nearest boundary. Using this distance map and given a user-defined width, during rendering, we are able to cull-off the voxels that are further from the surfaces than the given threshold, be they focus or context.

7. User interface

The application shows the graph and the clusters at the left side of a two-window widget (see Figure 4). The nodes are indicated with colored rectangular boxes. The edges are drawn with black arrows. The clusters are depicted as colored circles and the relationship $\mathcal{F}$ between nodes and cluster is shown with blue arrows. This side of the widget is editable: users can select nodes and modify their name and color using the options of the bottom part of the widget.

Users choose boolean operations and parenthesis on the menu bar at the top of the window and select nodes of the graph. The selected nodes form the focus of the visualization. They are shown at the right side of the widget with their parser computed color stored in LTF. This gives users a visual feedback on the subset of selected data. At the bottom of the widget, a ruler allows users establishing the level of required ghosting. When a non-zero ghosting level is selected, automatically, the context nodes are also shown in the right part of the widget. Moreover, the type of visualization of the context clusters can be specified by users by activating or not the *bounding boxes* flag and the *surface_voxels_only* flags. Zooming and panning can be performed at both sides of the widget, which is very useful for large graphs such as that of the *brain* dataset (see Section 8). The parameter β that modulates the influence of the LTF color in the shading is specified with a ruler at the bottom of the graphical area.

8. Implementation and results

We have implemented our method on top of the Tulip open-source graph visualization system [Aub03]. We have defined an aggregate class having as attributes a graph and the nodes properties. In order to accelerate the search of the selected clusters, we actually perform the graph traversal step and the mapping from nodes to clusters in one step. For this, we add as graph nodes the region identifiers. This way, the mapping step is implemented as another traversal level. The interface widget described in Section 7 uses Tulip visualization tools.

The parser is implemented with Bison GNU parser generator. We have defined a context-free grammar that describes the boolean operations between isoclasses. We have attached to every synthetic rule an action that calls a traversal procedure on the Tulip graph. We store the root nodes in a table and we use a hash table to have a direct access to the nodes.

Table 1: *Datasets: modality, size and website.*

Data	Mod.	size	http
foot	CT	128^3	curtard.org/~ andrew
brain	MR	181x217x181	sph.sc.edu
mouse	MR	256^3	idibaps.ub.edu
vhm	CT	512^2x1877	nlm.nih.gov

Table 2: *Graphs: number of nodes, edges and clusters; construction time in minutes; frames per second of rendering.*

Data	Nodes	Edges	Clusters	Const. mins	Render fps
foot	14	13	13	3,05	118
brain	138	602	116	87,2	70
mouse	26	384	120	92,7	35,6
vhm	27	53	10	80,3	2,2

We have tested our system on a Intel CoreDuo 6600@2400Mhz with 3.5 GB of RAM and a NVIDIA GeForce 8800 GTX with 768 MB of video memory. In order to test our system, we have used several biomedical datasets (see Table 1). For each dataset, different classification criteria, and therefore, different graphs can be constructed. Table 2 shows the characteristics of the graphs used in the color plates. The *foot* dataset represents a CT of a human foot. We have classified it into anatomical parts (each toe, the palm and the ankle) with slice-to-slice contour tracking tools. We have segmented it into soft tissue, bone and skin, and we have clipped it into left and right parts. *Brain* is a MR human brain already segmented into anatomical parts such as caudate and cerebellum, and left and right. For the *mouse* MR dataset of a laboratory mouse's brain we have used a co-registered SPECT model of the data and we have taken the activity level as a classification criterion in addition to clipping planes and anatomical labeling. Finally, the *vhm* comes from the Visible Human Male frozen CT repository. We have reconstructed the whole model and subdivided it into anatomical parts and segmented it into bone and soft tissue.

The construction time depends strongly on the size of the dataset and on the graph complexity. It is high for *mouse* and *brain* because of their graph complexity, and also for *vhm* because of its size. Rendering is very fast in all cases but slower in *vhm*, because this dataset is composed of 12 bricks and has the extra cost of loading several texture bricks per render. Figure 5 illustrates different queries and the effect of color ghosting. Figure 6 shows the difference between rendering all the context voxels or only their surface. Figure 7 shows the effect of the β parameter: the top image is rendered with TF only, while in the right-most image the weigth of LTF is of 0.75. The bottom image shows the surface of context with color ghosting. Figure 8 illustrates that our system can handle very large datasets such as the Visible Male. The videos attached to the submission (to reproduce with VLC) show how to use the system.

9. Conclusions

In this paper, we have presented a system for the illustrative visualization of structural relationships into multiclassified biomedical datasets. Our system provides new insights on the organization of data according to different taxonomies. It helps users to identify regions of the data that share common attributes, or have complementary characteristics.

Starting from this paper, we will continue our work in several directions. On one hand, we are currently working on designing a strategy to represent in the graph and render not only volume data but also polygon models of isosurface. On the other hand, the usability of the graph visualizer widget can be enhanced for the very complex graph structures by grouping nodes into metanodes and by allowing steerable exploration.

References

[ACS03] AXEN U., CARR H., SNOEYINK J.: Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications* (2003).

[Aub03] AUBER D.: Tulip : A huge graph visualisation framework. In *Graph Drawing Softwares*, Mutzel P., Jünger M., (Eds.), Mathematics and Visualization. Springer-Verlag, 2003, pp. 105–126.

[BE06] BRUCKNER S., E.GRÖLLER M.: Exploded views for volume data. *IEEE Trans. on Visualization and Computer Graphics 12*, 5 (2006), 1077–1084.

[BG05] BRUCKNER S., GRÖLLER M. E.: Volumeshop: An interactive system for direct volume illustration. In *IEEE Visualization* (2005), IEEE CS Press, pp. 671–678.

[BGKG06] BRUCKNER S., GRIMM S., KANITSAR A., GRÖLLER M. E.: Illustrative context-preserving exploration of volume data. *IEEE Trans. on Visualization and Computer Graphics 12*, 6 (2006), 1559–1569.

[BHW*07] BURNS M., HAIDACHER M., WEIN W., VIOLA I., GROELLER E.: Feature emphasis and contextual cutaways for multimodal medical visualization. In *EG/IEEE TCVG VisSym* (2007), pp. 275–282.

[CCS06] CHEN M., CORREA C., SILVER D.: Feature aligned volume manipulation for illustration and visualization. *IEEE Trans. on Visualization and Computer Graphics 12*, 5 (2006).

[CMH*01] CSEBFALVI B., MROZ L., HAUSER H., KÖNIG A., GRÖLLER E.: *Fast Visualization of Object Contours by Non-Photorealistic Volume Rendering*. Tech. Rep. TR-186-2-01-09, ICGA, Vienna UTech., 2001.

[FPT06] FERRÉ M., PUIG A., TOST D.: Decision trees for accelerating unimodal, hybrid and multimodal rendering models. *The Visual Computer 3* (2006), 158–167.

[HMBG01] HAUSER H., MROZ L., BISCHI G., GRÖLLER M.: Two-level volume rendering. *IEEE Trans. on Visualization and Computer Graphics 7*, 3 (2001), 242–252.

[KMH*02] KOSARA R., MIKSCH S., HAUSER H., SCHRAMMEL J., GILLER V., TSCHELIGI M.: Useful properties of semantic depth of field for better F+C visualization. In *EG/IEEE TCVG VisSym* (2002), pp. 205–210.

[KSW06] KRÜGER J., SCHNEIDER J., WESTERMANN R.: Clearview: An interactive context preserving hotspot visualization technique. *IEEE Trans. on Visualization and Computer Graphics 12*, 5 (2006).

[Lev91] LEVOY M.: Photorealistic volume rendering in scientific visualization. *SIGGRAPH C. Notes* (1991).

[LM04] LUM E., MA K.: Lighting transfer functions using gradient aligned sampling. In *IEEE Visualization* (2004), IEEE CS Press, pp. 289–296.

[LME02] LU A., MORRIS C., EBERT D.: Non-photorealistic volume rendering using stippling techniques. In *IEEE Visualization* (2002), IEEE CS Press., pp. 211–218.

[Nad00] NADEAU D.: Volume scene graphs. In *VVS '00: Proceedings of the 2000 IEEE Symposium on Volume Visualization* (New York, NY, USA, 2000), ACM, pp. 49–56.

[ONOT04] OWADA S., NIELSEN F., OKABE M., T I.: Volumetric illustration: Designing 3D models with internal textures. In *ACM SIGGRAPH* (2004), pp. 322–328.

[RBG07] RAUTEK P., BRUCKNER S., GRÖLLER M.: *Semantic Layers for Illustrative Volume Rendering*. Tech. Rep. TR-186-2-07-05, ICGA, Vienna UTech., 2007.

[RSK06] REZK-SALAMA C., KOLB A.: Opacity peeling for direct volume rendering. *Computer Graphics Forum 25*, 3 (2006), 597–606.

[SES05] SVAKHINE N., EBERT D. S., STREDNEY D.: Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics & Applic. 25*, 3 (2005), 31–39.

[SSBW05] STOCKINGER K., SHALF J., BETHEL W., WU K.: DEX: Increasing the capability of scientific data analysis pipelines by using efficient bitmap indices to accelerate scientific visualization. In *Scientific and Statistical Database Management* (2005), IEEE CS Press.

[Sto06] STONE M.: Color in information display. *IEEE Visualization course notes* (2006).

[VFSG06] VIOLA I., FEIXAS M., SBERT M., GRÖLLER E.: Importance-driven focus of attention. *IEEE Trans. on Visualization and Comp. Graph. 12*, 5 (2006), 933–940.

[VKG05] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven feature enhancement in volume visualization. *IEEE Trans. on Visualization and Computer Graphics 11*, 4 (2005), 408–418.

[WDC*07] WEBER G., DILLARD S., CARR H., PASCUCCI V., HAMMANN. B.: Topology-controlled volume-rendering. *IEEE Trans. on Visualization and Computer Graphics 13*, 2 (2007), 330–341.

[WEE02] WEISKOPF D., ENGEL K., ERTL T.: Volume clipping via per-fragment operations in texture-based volume visualization. In *IEEE Visualization* (2002), IEEE CSPress, pp. 93–100.

[WK95] WANG S. W., KAUFMAN A. E.: Volume sculpting. *ACM Interactive 3D Graphics* (1995), 151–156.

[WZMK05] WANG L., ZHAO Y., MUELLER K., KAUFMAN A.: The magic volume lens: An interactive focus+context technique for volume rendering. In *IEEE Visualization* (2005), IEEE CS Press, pp. 367–374.

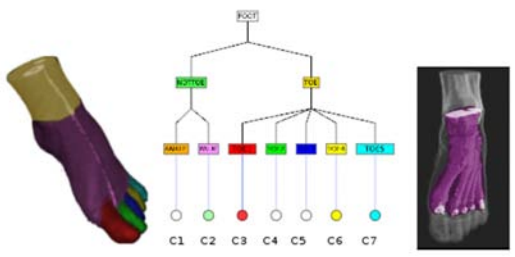
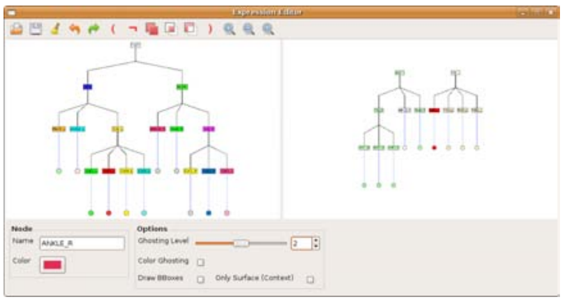


Figure 3: *Graph example. Left: the labelled voxel model in which each isoclass is rendered with a different color; Middle: the graph; Right: rendered for a given query.*



Node
Name ANKLE_R
Color

Options
Ghosting Level 2
Color Ghosting ☐
Draw BBoxes ☐ Only Surface (Context) ☐

Figure 4: *The selection interface widget. Left side: the graph; right side: the queried selection.*

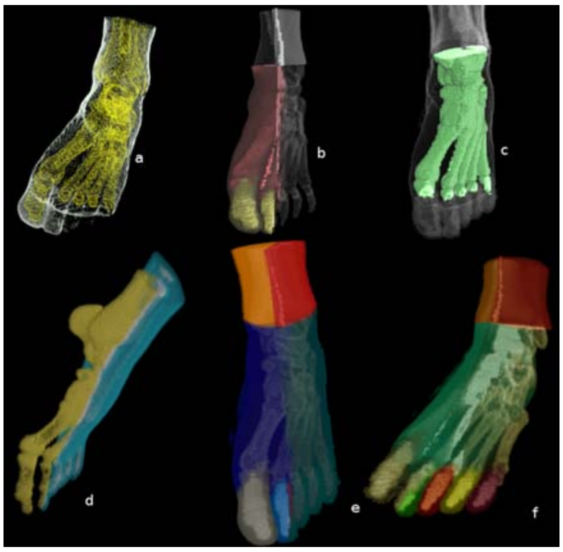


Figure 5: Foot *dataset. Queries:* ***(a)*** *Bone ∪ (Toe ∩ Right);* ***(b)*** *(Ankle ∩ Left) ∪ ((Toe ∪ Palm) ∩ Right));* ***(c)*** *Palm;* ***(d)*** *Bottom row, without color ghosting, Ankle - Left;* ***(e)*** *Ankle ∪((Toe ∪ Palm) ∩ Right;* ***(f)*** *Ankle ∪ Toe ∪ (Palm ∩ Right). Top row with color ghosting and bottom row without.*

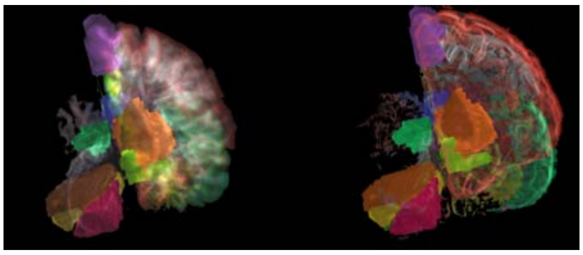


Figure 6: Brain *dataset. Query: (Left ∩ (Caudate ∪ Cuneous ∪ Putamen) ∪ (Right ∩ (Cerebellum ∪ Calcarine ∪ Crust)). Left: with inner context voxel; right: with context surface voxels only.*

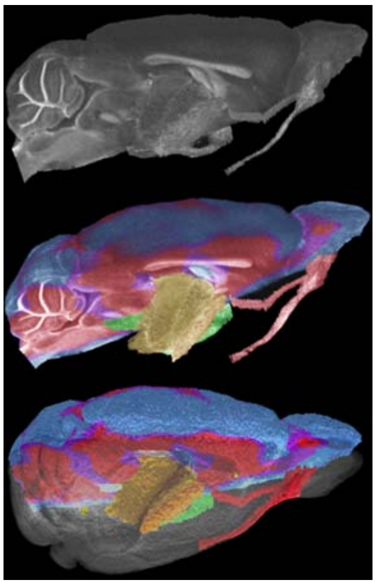


Figure 7: Mouse *dataset. Top: without ghosting,* β *= 0.0; Middle: without ghosting,* β*=0.75; Bottom: color ghosting, surface-only and* $\beta = 0.75$.

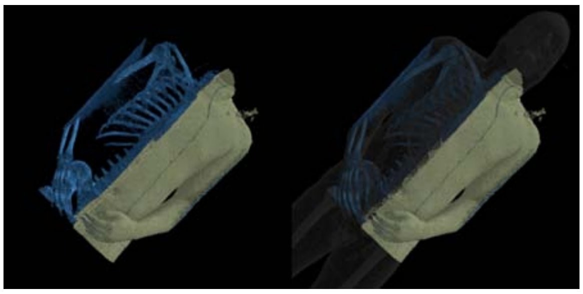


Figure 8: vhm *dataset. Query: Trunk ∩ (Bone ∪ (SoftTissue - Right));* β *= 0.40; Left without ghosting, Right with ghosting.*