Structural Focus+Context rendering of multiclassified volume data

P. Abellán and A. Puig and D. Tost

Abstract— We present a F+C volume rendering system aimed at outlining structural relationships between different classification criteria of a multiclassified voxel model. We clusterize the voxel model into subsets of voxels sharing the same classification criteria and we construct an auxiliary voxel model storing for each voxel an identifier of its associated cluster. We represent the logical structure of the model as a directed graph having as nodes the classification criteria and as edges the inclusion relationships. We define a mapping function between nodes of the graph and clusters. The rendering process consists of two steps. First, given a user query defined in terms of a boolean expression of classification criteria, a parser computes a set of transfer functions on the cluster domain according to structural F+C rules. Then, we render simultaneously the original voxel model and the labelled one applying multimodal 3D texture mapping such that the fragment shader uses the computed transfer functions to apply structural F+C shading. The user interface of our system, based on Tulip, provides a visual feedback on the structure and the selection. We demonstrate the utility of our approach on several datasets.

Index Terms—Volume Rendering. Structural information. Focus+Context

1 Introduction

Recently, as graphical hardware performance has increased, the interest of researchers to develop faster volume rendering methods has moved to the visual and semantical quality of the images. Today, a major concern in volume visualization is to provide meaningful images conveying relevant information on the data. On one hand, as datasets are each time larger, they contain more information than users can cope. On the other hand, precisely because of this data complexity, users have each time more difficulties to explore data with conventional interfaces. This is why, inspired in classical illustration techniques, new means of outlining significant features are being developed, such as cutaways and ghost views, silhouette edges enhancement, multiresolution Focus+Context (F+C) and adaptive shading.

A well known epistemologic principle is that the human mind organizes its knowledge of the world into taxonomies. It is our capacity of categorizing objects that gives us the ability of synthesizing the diversity. Information can be classified according to very diverse criteria, and the identification of data clusters sharing the same properties is essential for a good understanding of the data. Moreover, there are often logical implications between independent criteria that are important to detect because they represent the existence of hierarchies inside data. We are concerned with the visualization of volume data that have been classified in a pre-process according to several independent criteria. Our goal is to merge the different classification criteria in the visualization and to show the structure of the datasets providing users with structural F+C exploration tools. Existing illustrative visualization systems [29] [17] are useful for the exploration of preclassified datasets as well as non-classified ones. Some of them focus on multimodal and time-varying data [33], but none of them explicitly represent the structure of multiclassified datasets.

In this paper, we propose a data model and a rendering strategy capable of handling multiple classification criteria. Given a multiclassified voxel model, we represent the classification space as a directed graph such that the graph nodes represent classes and the graph edges logical implications between the corresponding classifiers. We set a boolean arithmetic on the basic classes and we construct an auxiliary voxel model that stores for each voxel an identifier of the combination of classification criteria that the voxel fulfills. Then, using the graph structure as a grammar, we design a parser that, given a visualization query expressed as a boolean expression of classes, creates several Transfer Functions (TFs) in the domain of the labelled voxel model. The rendering step is a multimodal 3D texture mapping that processes simultaneously the original voxel model and the labelled one. It uses the computed TFs to apply structural cut-away and ghosting effects that reveal the relationships between structures of the model.

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

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Previous work

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. We next survey these techniques.

Setting the camera so that it focuses on the relevant structures of the model requires sometimes a lot of user expertise. Therefore, many authors [3] [27] [29] [28] have addressed viewpoint optimization. Another technique to enhance structures of the volume is to simulate distortion lenses. This produces fish-eye like views that magnify features of interest [18] [9] [30]. However, viewpoint optimization and distortion lenses do not solve the problem that relevant structures of the volume can be occluded by others. For this, Bruckner et al. [4] proposed to automatically generate exploded views of a volume, that reveal the selection by moving away occluding parts. Alternatively, cutaway illustrations simply clip unwanted structures. Wang et al. [31] proposed to explore volumes using sculpting clipping tools. Weiskopf et al. [32] apply cutaway in 3D-texture mapping rendering. They describe *depth-based cutaway* with clipping geometry defined by simple convex boundary surfaces, and *texture-based volume clipping* where arbitrary clipping geometry is stored in an auxiliary texture used as mask. Owada et al. [22] describe an intuitive system that shows textured arbitrary cuts of surface models. Finally, Correa et al. [8] generalize the concept of virtual cuts and deformation by defining feature-aligned manipulation operators such as peelers, retractors and pliers.

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. It has proven to be an effective method to guide user's attention [16]. Depth-peeling has been used to make more transparent peripheral regions of the volume [10]. Ghost-

ing represents less-important regions with high transparency and relevant features with higher opacity [5]. Context surfaces can be outlined by modulating the opacity according to the gradient value [19] and by properly designing the TF [21]. Sparse representation of context is achieved by rendering only the contour of these structures [11]. Viola et al. [29] 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 and rendering techniques can be used for focus and context. Illustrative techniques such as pen-and-ink and stippling [20] can be used together with classical volume rendering techniques. Rautek et al. [23] introduce the concept of semantic layer that define the mapping of volumetric attributes to one visual style. A major novelty of this approach is that visual styles are described linguistically using semantic values.

Features can also be rendered separately with different algorithms and the resulting images can combined [14] [29]. The ClearView system [17] considers the context as composed of a set of pre-segmented regions or various iso-valued regions. It renders 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.[26] integrates boundary and silhouette enhancement and different shading styles, including toon. An interesting aspect of this system is that it appropriately selects the complexity of the illustrations on the basis of user's expertise level.

The systems described so far provide valuable illustrative images but, as far as we know, none of them addresses the visualization of multiclassified data. Doleish et al. [12] treat high-dimensional data resulting from computational simulation, which is somewhat similar to multiclassified data. They propose a Feature Definition Language that defines unions of features by brushing into multiple linked views of the data. They map each feature to a degree-of-interest (doi) that determines the color, opacity and size of the glyphs used in the visualization. Bruckner et al. [6] 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. They let users transform selection and provide a user control on the intersection regions via a 2D TF. Finally, Burns et al. [7] address illustrative visualization of an object of interest embedded into context volume data. They define a flexible cut-away structure that takes into account the importance of the structures defined in TF. They achieve very illustrative images merging an image of ultrasound with CT data. Nevertheless, their approach cannot model complex relationships between more than two modalities. Woodring et al. [33] address the combination of several volumes from a time-varying series into one rendered image. They define user queries as a tree structured boolean expression of operations between data-values or colors and they provide a visual feedback of the shading representing it as a tree. 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. Ferré et al. [13] propose to run-length codify combinations of data values in order to skip unselected regions during data traversal. Stockinger et al. [24] design a bitmap indexing scheme (DEX) that efficiently answers multivariate and multidimensional queries.

We base our work on the observation that even apparently simple datasets can have complex structural relationships. The analysis of data can benefit from an explicit modeling of this structure and from a user interface that allows users expressing data selections in terms of operations between structure components. Therefore, our work is related to the multimodal [13] and time-varying [24] approaches described above, but, by opposite to those, it explicitly models the relationships between structures. Besides, the representation of the structure of datasets with graphs and hierarchies in an active research area [15]. Multilevel algorithms to construct and visualize graph hierarchies and steerable exploration tools have been developed. We use previous research on that topic [1] for the design of our user interface.

3 Overview

Our rendering system helps users to explore the structural relationships between different classification criteria of datasets by providing Structural Focus+Context effects.

Figures 1 and 2 illustrate the pipeline of our method. First, in a pre-process, the initial voxel model is classified according to various independent criteria. This classification step can handle multimodal datasets such as MR/SPECT and MR/PET ones. We represent the classification space as a directed graph such that the graph nodes represent classes and the graph edges logical implications between the corresponding classifiers. The second step of our pipeline constructs this graph by clustering the voxels according to the class combinations and by identifying the dependencies between clusters. The output of this step is, in addition to the graph, a labelled voxel model that stores, for each voxel, a unique identifier of the cluster to which this voxel belongs.

The rendering stage is divided into two steps. In the first stage, users interactively express their rendering query as a boolean expression of the initial classification criteria. The query is the input of a parser that, using the graph structure, computes two transfer functions on the cluster labels domain. These transfer functions denoted as CTF and $GMTF$ indicate the visibility, color, opacity and gradient modulation factor of the different clusters. In the second step, rendering is performed by applying a multimodal 3D texture-mapping strategy that takes as input the original voxel model and the labelled one. The fragment shader fetches the voxel value and its label and, using these values, the corresponding rendering parameters needed for shading in the different transfer functions.

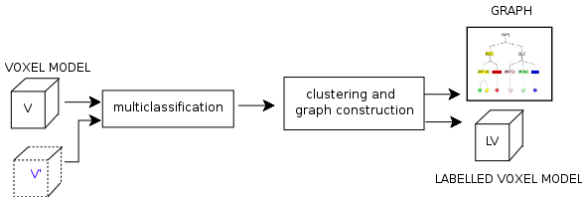

Fig. 1. Preprocess of the proposed method. The voxel model V is multiclassified, eventually taking into account other data modalities V'. The graph and the labelled voxel model LV are constructed.

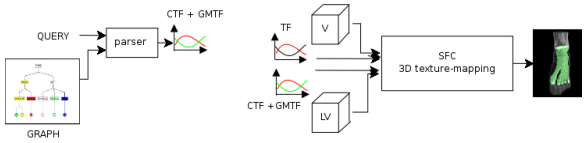

Fig. 2. Rendering step. Left: first step, given a user query and the graph, a set of transfer functions CTF and GMTF on the labelled voxel model is computed. Right: second step, multimodal 3D texture mapping handling simultaneously the original voxel model and the labelled voxel model together with their transfer functions.

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. We are able to visualize the structures than 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 Definition of the structure graph

Let V be the voxel model and nc be the number of independent classification criteria defined on V. Each classification criterion separates

the voxel model into a finite set of disjoint classes. We label each class using an integer identifier. We denote as $\vec{c_i}$ the set of class identifiers of the i-th criterion: $\vec{c_i} = \{c^i_1, \ldots, c^i_{nc_i}\}$, being nc_i the number of classes of the i-th criterion. We call *classification* function, the function $\mathscr{C}_i$ defined on the domain V and the codomain $\vec{c_i}$, that associates to every voxel $v \in V$ a unique class identifier: $\forall v, v \in V, \mathscr{C}_i(v) \in \vec{c_i}$. We define an isoclassified subset of V wrt to a class identifier c^i_j as the set of voxels of V having c^i_j as value of function $\mathscr{C}_i$: $isoclas(c^i_j) = \{v, v \in V, \mathscr{C}_i(v) = c^i_j\}$. Since the classification functions classify unequivocally all the voxels into disjoint classes, for every function $\mathscr{C}_i$, it is fulfilled that:

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

On the contrary, isoclasses of different criteria are not necessarily disjoint. Even more, one isoclass can include another one. It is precisely this type of relationship that we want to help users exploring through our application. 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 c^i_k isoclass:

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

We denote as $I = \{c^j_i, i \in \{1 \ldots nc\}, j \in \{1 \ldots nc_i\}\}$ the set of all the class identifiers and we call E the set all possible implication relationships between isoclass identifiers of I. Therefore, we represent the structure of V as a directed graph $G = (I,E)$ such that the nodes represent the isoclass identifiers and the edges implications between nodes. The direction of the edges is the opposite of the implication sense, thus the edge goes from inclusor isoclasses to included ones.

A simple example in Figure 3 illustrates these definitions. A CT foot dataset has been classified according to three different criteria. The first criterion has only two classes: *foot* or *empty*. The second criterion has four classes: *ankle*, *palm*, *toe* and *empty*. Finally the third criterion has 7 classes: *toe1* to *toe5*, *not toe* and *empty*. The *empty* classes are equivalent in the three criteria and have been removed from the graph. Therefore, the graph has 10 nodes. The inclusion relationships between isoclasses determine the existence of 9 edges depicted in blue in the figure. The graph is composed of a unique connected component and has a depth of 3.

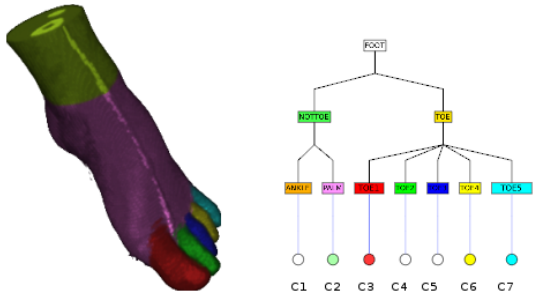

Fig. 3. Graph example. Left: the labelled voxel model in which each isoclass is rendered with a different color; Right: the graph

4.2 Definition of the labeled voxel model

As explained in Section 3, during the graph construction step, we compute a labelled voxel model LV that we use as a mask to cut-away and to ghost isoclassified regions of V. The definition of this voxel model and its relationship with graph G relies on the concept of *cluster* that we next introduce.

The identifiers of the isoclasses to which a voxel v belongs is an nc-tuple of class identifier values $\vec{C(v)} = (C_1(v), \ldots, C_{nc}(v))$. During the process of graph construction, we clusterize the voxels into sets of voxels sharing the same isoclass identifiers tuple. We call *clusters* each of these sets and, given a cluster r, we denote as $\vec{C_r}$ the nc-tuple of its associated class identifiers. Specifically, a cluster r is a non-empty subset of V such that: $\forall v \in r : \vec{C(v)} = C_r$. 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: $R = \{r_i, i \in \{1 \ldots nr\}\}$. Again, it is fulfilled that all the clusters are disjoints and their union is the voxel model:

- $\forall i,j : 1 \leq i,j \leq nr : r_i \cap r_j = \emptyset$
- $\bigcup_{j=1}^{nr} r_i = V$

We define the labelled voxel model LV, as a voxel model with the same resolution and spatial orientation than V and such that the value of every voxel in LV is the identifier of the cluster to which the voxel belongs: $\forall v, v \in LV : value(v) = r / r \in R$. Finally, we define a binary relation $\mathscr{F}$ with domain the isoclass identifiers set I and with codomain the clusters set R, that associates to a class identifier of $i \in I$ a cluster $r \in R$, if $i \in C_r$ and no descendant of I in the graph G has an identifier in C_r. More formally: $\forall i \in I \wedge \forall r \in R, i \mathscr{F} r$ `iff` $\exists c_i \in C_R / i = c_i \wedge \neg(\exists c_j \in C_R, c_j \neq c_i / c_j \Rightarrow c_i)$. We use $\mathscr{F}$ in order to access to the voxels in LV during the graph G traversal on users query.

In the example of Figure 3, there are 7 clusters (c1..c7) depicted with colored circles. The blue arrows depict the relationship $\mathscr{F}$ between a node of the graph and a cluster.

4.3 User queries and parser rules

In order to explore the structure of the dataset, we want to provide users with tools allowing them to select union, intersections and complementary sets of the isoclasses. We express user queries as boolean expressions defining the isoclasses to which selected voxels belong. Specifically, let $b^i_k(v)$ be a boolean expression that indicates if voxel $v, v \in V$ belongs to the isoclass identified by c^i_k: $b^i_k(v) \Leftrightarrow v \in isoclass(c^i_k)$. A user query can be expressed as a combination of b^j_i 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 $\mathscr{M}$ the function that associates to any user query a unique union of clusters in R. It is based on the following rules, for all $i,j \in \{1 \ldots nc\}$, $k, k1, k2 \in \{1 \ldots nc_i\}$ being $k1 \neq k2$, and $l \in \{1 \ldots nc_j\}$:

- $\mathscr{M}(b^i_k) = \bigcup_{m=1}^{nr} r_m$ such that $c^i_k \in C_{r_m}$
- $\mathscr{M}(\neg(b^i_k)) = \bigcup_{m=1}^{nr} r_m$ such that $c^i_k \notin C_{r_m}$
- $\mathscr{M}(b^i_k \cup b^j_l) = \bigcup_{m=1}^{nr} r_m$ such that $c^i_k \in C_{r_m} \vee c^j_l \in C_{r_m}$
- $\mathscr{M}(b^i_{k1} \cap b^i_{k2}) = \emptyset$
- $\mathscr{M}(b^i_k \cap b^j_l) = \bigcup_{m=1}^{nr} r_m$ such that $c^i_k \in C_{r_m} \wedge c^j_l \in C_{r_m}$

The parser implements these rules and traverses the graph until reaching nodes that are $\mathscr{F}$-related to clusters. Throughout the traversal, it computes the list of bounding boxes that must be rendered and two transfer functions for the labelled voxel model LV: the cluster Color Transfer Function CTF and the cluster Gradient Modulation Transfer Function $GMTF$. These transfer functions store rendering parameters following the structural F+C 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 isoclasses number of voxels and their bounding box. We set the opacity of the isoclass as a factor of their bounding sphere radius and their number of voxels. We compute the color of each isoclass according to Stone's coloring model [25], but we also let users interactively ediding them at their convenience. We create one graph node per isoclass. Then, we compute for each isoclass

the set of isoclasses that it contains. To do so, we process the input voxels slice by slice in order to avoid an excessive memory load for huge models and high number of classification criteria. 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.

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 isoclasses identifier. We label the voxel with this identifier in LV and, if the cluster identifier is new, we insert it in the cluster list. The relationship $\mathscr{F}$ between the graph nodes and the clusters is codified in the hash code.

6 Rendering

The rendering step is based on multimodal 3D texture-mapping. It renders and compose a set of proxy polygons parallel to the viewing plane. 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 functions CTF and $GMTF$ computed by the parser and codified as 1-D textures. The property value r is used to fetch the voxel color and opacity in the transfer function of V. The fragment shader combines these values in the shading equation, getting structural cut-away, ghosting and coloring effects.

After texture slicing, all the bounding boxes computed by the parser are rendered in wireframe.

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 CTF transfer function. In the rendering step, the clusters colors are blended with the voxels colors according to a user-specified blending factor $beta$, $0 \leq \beta \leq 1$.

The computation of CTF is 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 $r = \mathscr{M}(\neg b^i_k)$ then $CTF(r) = \overline{col(b^i_k)}$
- if $r \in \mathscr{M}(b^i_k \cup b^j_l)$ then $CTF(r) = col(b^i_k) \cup col(b^j_l)$
- if $r \in \mathscr{M}(b^i_k \cap b^j_l)$ then $CTF(r) = col(b^i_k) \cap col(b^j_l)$

6.2 Structural Cut-away

Cut-away is achieved through the definition of the α-channel of the *Color Cluster Transfer Function (CTF)*. 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.

6.3 Structural ghosting

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 CTF_C. This computation is done after the user query has been evaluated and the focus clusters have been determined. Starting from the graph nodes $\mathscr{F}$-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 in the same way as that of the focus, applying structural coloring, as described in Section 6.1. Alternatively, if a *color_ghosting* flag is activated, the parser can set the values of CTF 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.

6.4 Shading

The shader computes surface shading (Phong's model), emission and absorption (E+A) or both models (E+A+S) 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 $GMTF$ computed by the parser. 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 CTF, to prevent context voxels from being rendered during texture slicing.

Figure 4 summarizes the fragment shader instructions.

6.5 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 define 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 5). The nodes are indicated with colored rectangular boxes. The edges are drawn with blue arrows. The clusters are depicted as colored circles and the relationship $\mathscr{F}$ between nodes and cluster is shown with black arrows. This side of the widget is editable: users can select nodes and modify their name and color using the edition options of the bottom part of the widget.

```
uniform sampler3D volume;
uniform sampler3D mask;
uniform sampler1D lut;
uniform sampler1D CTF;
uniform sampler1D GMTF;
uniform bool surface_voxels_only;
uniform bool color_ghosting;
uniform float beta;
void main(void)
{
    r = texture3D(mask,gl_TexCoord[1].stp);
    col_mask = texture1D(CFT, r.a);
    if(col_mask.a == 0.0) discard;
    else {
       p = texture3D(volume,gl_TexCoord[0].stp);
       col_vol = texture1D(lut, p.a);
       grad = p.rgb;
       grad_weight = texture1D(GMTF, r.a);
       if(col_mask.a ==1.0)
            color.rgba = col_vol.rgba*beta+ (1-beta)*col_mask.rgba;
       else {
            if(surface_voxels_only && !LargeGradient(grad, grad_weight))
              color.a = 0.0;
            else
              if (color_ghosting) color.rgba = col_mask.rgba;
              else color.rgba = col_vol.rgba*beta+ (1-beta)*col_mask.rgba;
       }
       gl_FragColor = shade(color, grad, grad_weight);
    }
}
```

Fig. 4. Fragment shader pseudocode

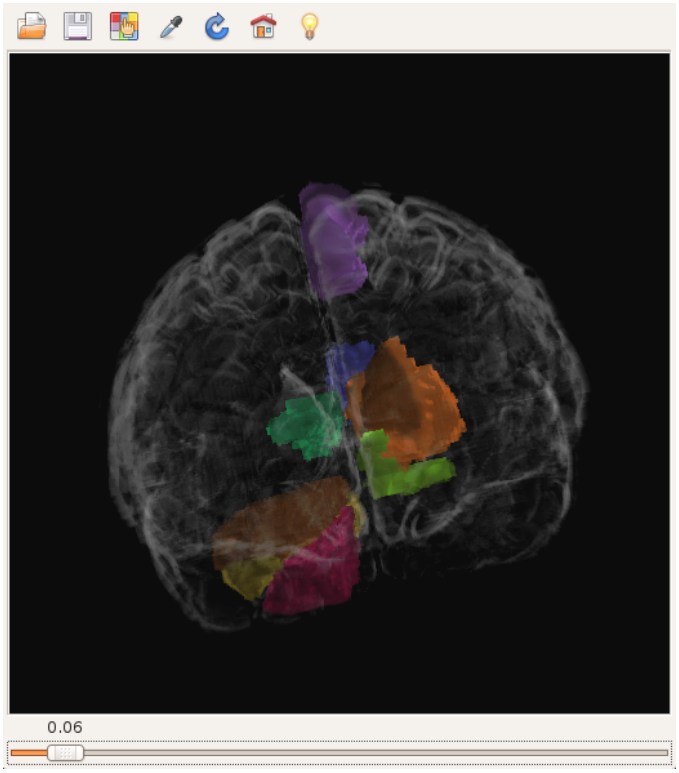

Fig. 6. Rendering of the *brain* dataset. Several structures inside the brain are on focus. Ghosting color, maximum ghosting level and surface_voxels_only are set with a very low β value that makes visible only the boundary surface of the context.

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 CTF. This gives users a visual feeback 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 Figure 7. The parameter β that modulates the influence of the CTF color in the shading is specified with a ruler at the bottom of the graphical area, as shown at the bottom of Figure 6.

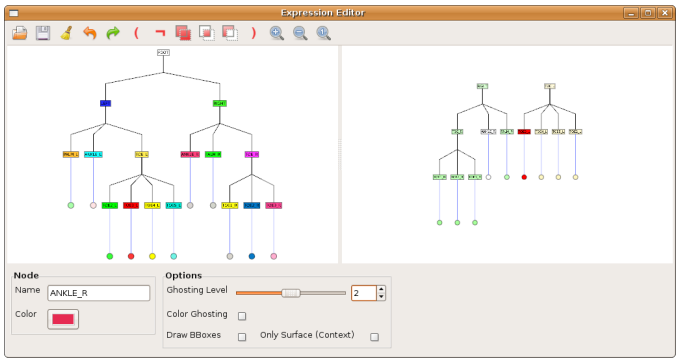

Fig. 5. The selection interface widget. Left side shows the graph and right side the selection.

8 Implementation and Results

We have implemented our method on top of the Tulip graph visualization system [2]. 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 graph visualization tools.

We have designed our parser using Bison GNU parser generator. For this, we have defined a context-free grammar that describes the boolean operations between isoclasses as exposed in Section 4.3. We have attached to every synthetic rule an action that calls a traversal procedure on the Tulip graph. We use a hash table to have a direct access to the graph nodes and start the traversals directly at the involved nodes. In addition, we store into a table the root nodes of the graph. This grammar is taken as input to Bison to generate the parser.

In order to test our system, we have used several datasets (Table 8). The *foot* and the *lego* are from Andrew Winter's repository (http://www.curtard.org/ andrew), the *porshe* is from the Volume Library (http://ww9.informatik.uni_erlangen.de) and the *brain* comes with the MRIcro system (http://www.sph.sc.edu).

For each dataset, different classification criteria, and therefore, different graphs can be constructed. For instance, the graph of the foot used for the simulations is more complex than the one used in Section 4. We got the *brain* dataset already classified according to the different anatomical parts that compose a brain. We have segmented and classified the other datasets in the preprocessing stage according to three different methods: clipping planes to differentiate the left and right, up and down regions; value-based segmentation to classified isorange regions such as, bone versus skin or wheel versus chassis, and contour tracking to separate regions within the same range of values such as the different toes from the palm and the ankle from the foot and the

different pieces of the legoman.

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. Table 8 shows the characteristics of the constructed graphs. Table 8 indicate the time needed for the graph construction and for rendering. The duration of the construction step depends on the number of isoclasses. As shown in Figure 7, the *brain* dataset has a complex structure, therefore its construction is slower than the others. The rendering time is expressed in frames per second (fps). We have computed the fps as the average fps for various rendering with the camera moving around the data. The rendering is very fast, it provides interactive rates for all datasets. The differences between the rendering time by varying the selection and the ghosting level are almost insignificant, because the datasets are not very large and fit into memory.

Table 1. Datasets size

foot	lego	porshe	brain
128x128x128	171x98x244	128x128x128	181x217x181

Table 2. Graph characteristics

Dataset	Levels	Nodes	Edges	Clusters
foot	4	14	13	13
lego	4	16	15	11
porsche	4	30	29	22
brain	5	138	602	116

Table 3. Construction and rendering time

	foot	lego	porshe	brain
construction	3.05 min	7.23 min	9.32 min	1h52
rendering	118 fps	135 fps	85 fps	76fps

Tables 4 to 7 show different images of the datasets that illustrate the structural ghosting and cut-away concepts.

The images of the *foot* dataset show that specific nodes of the graph can be selected while the whole structure of the foot is still visible. The β parameter influences the visualization, as well as the TF of the original voxel model. For instance, in the upper left image of Table 4, β has value 1.0, therefore the CTF is not taken into account in the shading. All the foot structures but the four small toes have been selected. Ghosting is applied and, since only surface voxels are rendered, the smaller toes appear lighter than the larger ones. In the other images β is smaller than 1.0, so the CTF influences the visualization. In the upper right image, there is no color ghosting so the context palm is shown in yellow. In the other images, the context is white because color ghosting has been enabled. The images of the *lego* and the *porshe* dataset show different selection of the focus, all with ghosting. Finally, the *brain* images shows selections on features of the two hemispheres and the cerebellum of the brain contextualized with different levels of ghosting.

9 Conclusions

In this paper, we have presented a system for the illustrative visualization of structural relationships into multiclassified 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, on the contrary, have complementary characteristics.

Starting from this paper, we will continue our work in several directions. On one hand, we want to analyze how our to adapt our structure

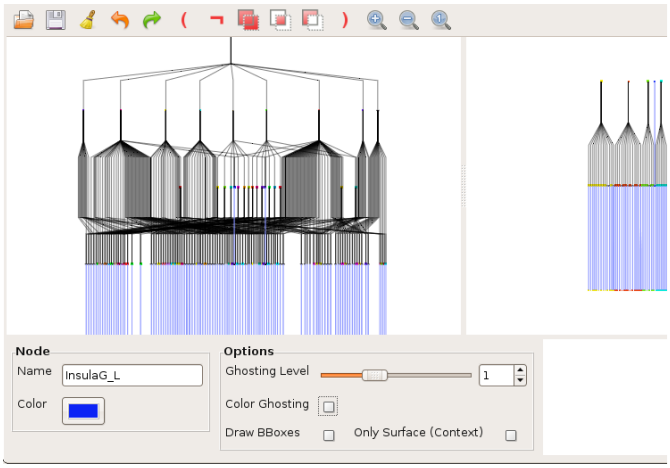

Fig. 7. Graph of the *brain* dataset.

to very huge datasets split into several voxel models at several resolution levels. On the other 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. Finally, 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 as proposed by Archambault et al. [1].

References

[1] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multi-level graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[2] D. Auber. Tulip : A huge graph visualisation framework. In P. Mutzel and M. Jünger, editors, *Graph Drawing Softwares*, Mathematics and Visualization, pages 105–126. Springer-Verlag, 2003.

[3] U. Bordoloi and H.-W. Shen. View selection for volume rendering. In *IEEE Visualization*, page 62, 2005.

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

[5] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Trans. on Visualization (to appear)*, 12(6):1559–1569, 2006.

[6] S. Bruckner and M. E. Gröller. Volumeshop: An interactive system for direct volume illustration. In H. R. C. T. Silva, E. Gröller, editor, *IEEE Visualization 2005*, pages 671–678, Oct. 2005.

[7] M. Burns, M. Haidacher, W. Wein, I. Viola, and E. Groeller. Feature emphasis and contextual cutaways for multimodal medical visualization. In *Eurographics / IEEE VGTC Symposium on Visualization 2007*, pages 275–282, May 2007.

[8] M. Chen, C. Correa, and D. Silver. Feature aligned volume manipulation for illustration and visualization. *IEEE Trans. on Visualization and Computer Graphics*, 12(5), sep - oct 2006.

[9] M. Cohen and K. Brodlie. Focus and context for volume visualization. *Theory and Practice of Computer Graphics 2004 (TPCG'04)*, 00:32–39, 2004.

[10] C.Rezk-Salama and A. Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum (Proc. Eurographics)*, 25(3):597–606, 2006.

[11] B. Csebfalvi, L. Mroz, H. Hauser, A. König, and E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. Technical Report TR-186-2-01-09, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Apr. 2001. human contact: technical-report@cg.tuwien.ac.at.

[12] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *IEEE TCVG-EG VisSym 2003*, pages 239–248, 2003.

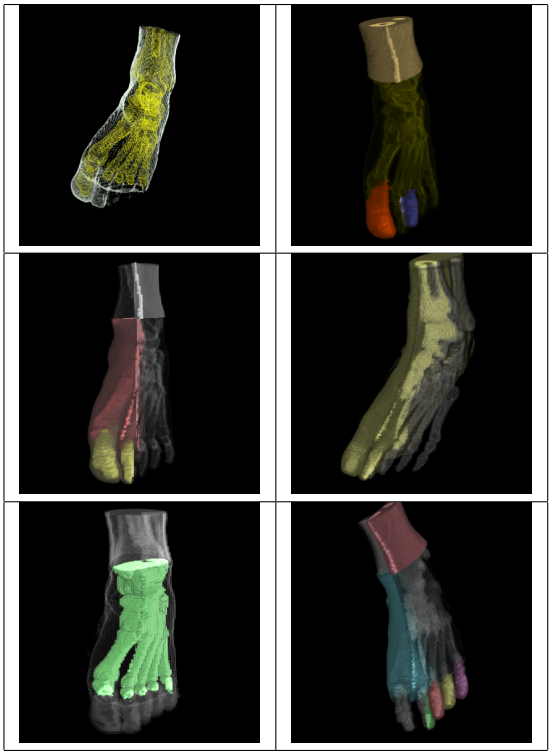

Table 4. *Foot* results with different levels of ghosting and different selections.

Table 5. *Lego* results with different levels of ghosting and different selections.

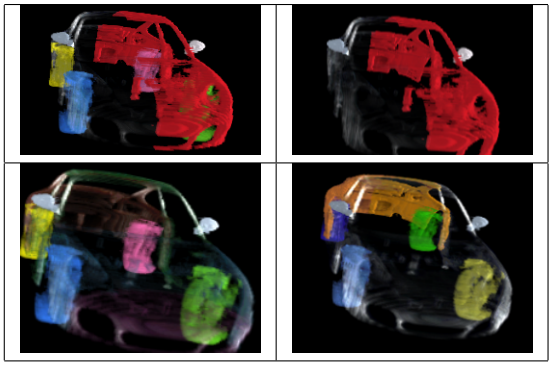

Table 6. *Porsche* results with different levels of ghosting and different selections.

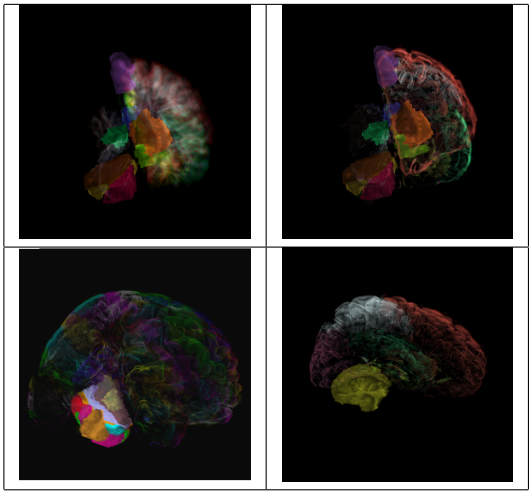

Table 7. *Brain* results with different levels of ghosting and different selections.

[13] M. Ferré, A. Puig, and D. Tost. Decision trees for accelerating unimodal, hybrid and multimodal rendering models. *The Visual Computer*, 3:158–167, 2006.

[14] H. Hauser, L. Mroz, G. Bischi, and M. Gröller. Two-level volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[15] I. Herman, G. Melancon, and M.Marshall. Graph visualization and navigation in information visualization: a survey. *IEEE Trans. on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[16] R. Kosara and al. Useful properties of semantic depth of field for better f+c visualization. *Proc. visSym 2002*, pages 205–210, 2002.

[17] J. Krüger, J. Schneider, and R. Westermann. Clearview: An interactive context preserving hotspot visualization technique. *IEEE Trans. on Visualization and Computer Graphics (Proc. Visualization / Information Visualization 2006)*, 12(5), sep- oct 2006.

[18] E. LaMar, B. Hamann, and K. Joy. A magnification lens for interactive volume visualization. *Proc. Ninth Pacific Conf. on CG&App.*, pages 223–, 2001.

[19] M. Levoy. Photorealistic volume rendering in scientific visualization. *SIGGRAPH 91 Course Notes*, 1991.

[20] A. Lu, C. Morris, and D. Ebert. Non-photorealistic volume rendering using stippling techniques. In *Proc. of the conference on Visualization '02*, pages 211–218. IEEE Press, 2002.

[21] E. Lum and K. Ma. Lighting transfer functions using gradient aligned sampling. *vis*, 00:289–296, 2004.

[22] S. Owada, F. Nielsen, M. Okabe, and I. T. Volumetric illustration: Designing 3D models with internal textures. *Proc. of ACM SIGGRAPH*, pages 322–328, 2004.

[23] P. Rautek, S. Bruckner, and M. Gröller. Semantic layers for illustrative volume rendering. Technical Report TR-186-2-07-05, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2007.

[24] K. Stockinger, J. Shalf, W. Bethel, and K. Wu. Dex: Increasing the capability of scientific data analysis pipelines by using efficient bitmap indices to accelerate scientific visualization. In *International Conference on Scientific and Statistical Database Management (SSDBM), Santa Barbara, California, USA*. IEEE Computer Society Press, June 2005.

[25] M. Stone. Color in information display. *IEEE Visualization course notes*, 2006.

[26] N. Svakhine, D. S. Ebert, and D. Stredney. Illustration motifs for effective medical volume illustration. *IEEE Comput. Graph. Appl.*, 25(3):31–39, 2005.

[27] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. *vis*, 0:63, 2005.

[28] I. Viola, M. Feixas, M. Sbert, and E. Gröller. Importance-driven focus of attention. *IEEE Trans. on Visualization and Computer Graphics*, 12(5):933–940, Oct. 2006.

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

[30] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman. The magic volume lens: An interactive focus+context technique for volume rendering. In

Vis05:Proc. of the conference on Visualization '05, pages 367–374. IEEE Press, 2005.

[31] S. W. Wang and A. E. Kaufman. Volume sculpting. *ACM Symp. on Interactive 3D Graphics*, pages 151–156, 1995.

[32] D. Weiskopf, K. Engel, and T. Ertl. Volume clipping via per-fragment operations in texture-based volume visualization. In *Proc. of the conference on Visualization'02*, pages 93–100. IEEE Press, 2002.

[33] J. Woodring and H. Shen. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):909–916, 2006.