Design of a Multimodal Rendering System

Anna Puig, Dani Tost and Maria Ferré
UPC Software Dept.
URV CS and math Dept.

Abstract

This paper addresses the rendering of aligned regular multimodal datasets. It presents a general framework of multimodal data fusion that includes several data merging methods. We also analyze the requirements of a rendering system able to provide these different fusion methods. On the basis of these requirements, we propose a novel design for a multimodal rendering system. The design has been implemented and proved showing to be efficient and flexible.

1 Introduction

Volume Graphics is today used in many different areas of Scientific Visualization and Realistic Image Synthesis. Various popular volume visualization toolkits and applications exist (VolVis [Avila et al. 1994], VTK[Schroeder et al. 1998] and VolPack [Lacroute 1995] among others) and many surface-based systems have been extended to volume data [Eckel 1999]. However, most of this research and development focuses at monoproperty and monomodal data. Modeling and rendering various properties of the same 3D region, measured with different registration devices (multimodal studies) or at different instants of time (dynamic studies) have been less addressed in the bibliography [Zuiderveld et al. 1996; Cai and Sakas 1999]. Nevertheless, the demand for multimodal systems is increasing in many applications. In the medical field, for instance, the combination of data from MRI (Magnetic Resonance Imaging), CT (Computer Tomographies), SPECT (Single Photon Emission Computed Tomography) and PET (Positron Emission Tomography) provides a better perception of the anatomy and its relationship with the functional activity, a key point to detect pathologies and understand behaviours.

A large number of papers on multimodality address the so-called *registration problem* [Lemoine et al. 1991], [Pelizzari et al. 1989], [van der Elsen 1993], [Ferre and Tost 2001]. The registration consists of computing a set of rigid or elastic transformations that align the datasets in a common reference system. This process is necessary when the different property sampling have not been realized simultaneously. After the registration transformation has been computed, the datasets are generally resampled at the same resolution in the common reference system. This introduces sampling errors and it may dramatically increase memory requirements when the initial resolution of the datasets is very different. However, keeping the datasets in their original system and applying during rendering the geometrical transformations computed in the registration step would increase the visualization cost. In addition, it is not feasible in all rendering strategies. It is straightforward in ray-casting, less easy in splatting by sheets perpendicular to the viewing direction and not possible in ordered traversals. In this paper, we assume that the initial models are regular, aligned and that they have the same resolution. The volume representations used are voxel models.

Other references focus at multimodal rendering itself. They can be classified into two categories: 2D and 3D methods. The former ones [Hawkes et al. 1990], [Spetsieris et al. 1995] perform the merging slice-to-slice. Either they visualize the slices in parallel windows or they interlace image lines or they merge color values. Three dimensional methods perform the integration of the 3D data. In general, they give a better perception of the spatial relationships between data than 2D techniques. However, according to [Hu et al. 1990], the interpretation of merged data is sometimes easier on 2D images that are simpler and free from opacity accumulation effects. 3D integration can thus provide means of locating areas of interest that can afterwards be sliced and studied in 2D. In addition, 3D techniques can profit from fitted surface rendering effects. As an example, the "Normal fusion" strategy proposed in [Zuiderveld et al. 1996] shows SPECT values below a brain surface fitted into MR data. Other relevant previous work on 3D techniques are [El-Khalili et al. 1996] and [Cai and Sakas 1999], both based on ray-casting through volume data. The first paper analyzes the fusion of CT and SPECT data and the second one CT, segmented CT and radiotherapy dose. Both papers propose two types of integration: object-order merging, i.e. merging data before projection and image-order merging, i.e. intermixing images.

The goal of this paper is to define a general framework for multimodal rendering that could include different 3D fusion techniques and adapt them to image-order rendering techniques such as ray-casting as well as object-order ones such as splatting and texture mapping. A classification of different fusion modes according to the step in the rendering pipeline at which fusion occurs is proposed in Section 2. In order to implement these techniques, the requirements of a multimodal system are defined in Section 3. Current volume applications are not easily adaptable to fit these requirements. Therefore, we have designed a multimodal system that fulfills the requirements and that can be used as a workbench to test the different techniques. This design is discussed in Section 4. In section 5, an implementation of it is described and some simulations results are shown.

2 Multimodal rendering methods

In multimodal visualization, the rendering pipeline for any sample point or voxel, takes as input the set of n property values and it outputs one color and opacity. This color and this opacity may be computed in two different ways:

- rendering only one property per point (OPP scheme One-Property-per-Point), i.e. selecting at each sample only one of the properties,
- rendering simultaneously various properties per point (MPP scheme, Multiple-Properties-per-Point), i.e. actually mixing properties by weighting them.

The selection of one element as well as the mixture of elements may be realized at different steps of the rendering pipeline. Therefore, we distinguish five fusion modes according to the parameters that drive the fusion [Ferre et al. 2002]: PF (Property Fusion) based on property values, P&GF (Property & Gradient Fusion) driven by property and gradient values, MF (Material Fusion) that uses materials, SF (Shading fusion) also based on materials but that realizes the fusion within the shading computation and, finally, CF (Color Fusion) based on final colors.

These different fusion methods rely on *decision functions* that, given a combination of properties, gradients, materials or colors, either choose one of them (OPP scheme) or compute a set of weights that should be applied to the input parameters in order to mix them in a linear combination or eventually using higher order functions. Being $p_1, p_2, .. p_n$ the set of n input parameters, the decision functions can be defined as:

$$f_{opp}(p_1, p_2, .., p_n) = p_i, 1 \leq i \leq n \quad (1)$$

$$f_{mpp}(p_1, p_2, .., p_n) = (w_1, w_2, .. w_n) \quad (2)$$

The decision functions are based on the evaluation of boolean expressions relative to the ranges of value of the different parameters. The chosen parameter or the weight arrays that the decision function returns depend on the combination of range values in which the input parameters fall.

The two fusion schemes and five fusion modes lead to the nine following multimodal rendering algorithms:

- Method 1: Property fusion (PF) and OPP scheme. Given a sample point or voxel, n property values are computed. The decision function selects one of them and the rendering pipeline goes on as in a monomodal study. This method is applicable to any combination of datasets.

- Method 2: Property fusion (PF) and MPP scheme. This mode is usable only when the datasets represent the same physical property. An example of this case is a multimodal study based on CT and MR which both sample tissue density at a precision of one byte per value. The n property values at a sample point or voxel are used in the decision function to compute a set of n weights. The resulting property is calculated as the weighted sum of the input properties. Next, this value is used in the classification using global transfer functions defined for the multimodal study. The gradient vector is calculated by weighting the individual gradients. Finally, shading is realized as usual.

- Method 3: Property and Gradient Fusion (P&GF) and OPP scheme. This method is similar to method 1, but it takes into account the gradient values in the decision function. The definition of this function and its evaluation are more complex, but it permits to better outline isosurfaces of some of the properties.

- Method 4: Property and Gradient Fusion (P&GF) and MPP scheme. This method resembles the second one but using gradient information like method 3.

- Method 5: Material Fusion (MF) and OPP scheme. Starting from n property values, n classification processes are done separately. Then, the decision function selects one of the materials and shading is computed using this material. The gradient vector is calculated as the average of all the gradient vectors of properties which have been classified as the selected material. By comparison to mode 1 and 3, MF effects are less tinged but it is more intuitive and easy to specify for the users.

- Method 6: Material Fusion (MF) and MPP scheme. This method weights n materials and uses them for shading with a gradient vector computed as the weighted sum of the initial gradient vectors. It requires the materials to be of the same type in order to mix them. As method 5, it is less flexible than methods 4 and 2 but easier to define.

- Method 7: Shade Fusion (SF) and MPP scheme. This method is similar to the previous one, but instead of averaging materials and gradients before shading, it uses them with their computed weights directly in the shading equation. In comparison to method 6, this method outlines better the effect of light sources on each fitted surface. When no surface shading is done, it gives the same results as method 6.

- Method 8: Color Fusion (CF) and OPP scheme. This method performs n complete rendering pipelines and the decision function selects one of them. It is the simplest method and, when applied to pre-shaded RGBα models, it is the fastest. However, it is difficult to tune, specially if the original properties map onto the same color space.

- Method 9: Color Fusion (CF) and MPP scheme. This method is similar to the previous one, but it weights colors instead of choosing one.

Color Plates 1 to 9 show examples of these methods. The images are explained in Section 5.

3 Requirements

A system able to process multimodal rendering methods such as the ones proposed in the previous section should fulfill different types of requirements: (i) conditions on the type of data and material that should be supported, (ii) functional requirements on rendering, classification, shading and fusion algorithms, (iii) general requisites.

Many of these requirements apply for monomodal data as well as for multimodal ones. However, in the latter case it should be feasible to handle the variety of possibilities simultaneously.

The requirements of the data are the following:

- Object domain. The system should handle various types of objects, surface representation as well as volume models.

- Property domain. For volume models, different types of properties should be handled, such as gray density values with different number of precision bits, RGB colors, floats and arrays of floats.

- Gradient computation methods. Several gradient computation methods should be provided as, for instance, forward, backward and central difference for one-dimensional property values and per-component o per-norm for multi-dimensional property values.

- Optical property domain. Each data set may represent different types of materials, i.e. optical properties and textures of surfaces (reflection coefficients, diffuse and specular colors, transmission coefficient, etc), optical properties and textures of volume (emission, absorption and scattering coefficients) and hybrid volume and surface data properties.

In relation to algorithms, the conditions to be fulfilled are:

- Rendering method : Several rendering method should be available: wireframe, z-buffer, ray-casting, by-sheet splatting, sorted traversal, shear-warp and texture mapping. Each object may eventually have its own rendering method in order to compose local visualizations in a one image. In addition, each object may have its own camera and light sources. Having per-object camera is useful to apply different levels of zoom to different objects. Per-object lighting permits to apply light sources only to some chosen objects, a non-realistic effect that can however be interesting to outline some features of these objects.

- Classification methods. Several mechanisms to compute the material or combination of materials existing at a sample point inside the volume should be provided: from transfer functions to explicit look-up tables or more complex probabilistic classification schemes. It should be noted that classification is more complex in multimodal studies, because, it may require *n*-parameterized transfer functions, being *n* the number of properties multiplied by their dimension. The implementation of these functions via look-up table, which is usually faster than other methods, may be infeasible in multimodal studies, because of its huge memory occupancy.

- Shading function. The shading function computes the color intensity contribution at a sample point. Different shading strategies are required. First, the *value shading* that directly maps the value to a gray or RGBα color scale. Next, the *color shading* that maps the value to color using a color table. Finally, the actual shading functions, that use the material optical properties and eventually the gradient value: *emission-only shading*, *emission+absorption shading* and *Phong shading*. In addition, other parameters may be activated such as depth cueing, light sources attenuation and constant opacity.

- Fusion With regard to fusion, the two different fusion schemes (OPP and MPP) combined with the five fusion modes (PF, P&GF, MF, SF and CF) described in Section 2 should be provided. In addition, the decision functions should be easily modifiable during the application execution.

Finally, the general requisites are (i) the everlasting need for efficiency in the operations, especially per-voxel queries which are often the bottleneck of volume management (ii) the flexibility for the user to change any parameter during execution.

Very important from a programmer's point of view is also the easiness of code writing, maintenance and extensibility. These two aspects are discussed with more depth in the design section.

4 Design

Several volume rendering applications and libraries are currently available. Up to our knowledge, they are mostly oriented at rendering monomodal data. Most of them support different property data types. The Visualization ToolKit (VTK) [Schroeder et al. 1998], for instance, handles single valued gray-scales, rgb and 3x3 tensors among others. VolPack [Lacroute 1995] permits variable number of bits per property and thus, it implicitly supports different types of property. Property Fusion (PF) can be simulated in any of these systems by constructing a new merged model in a preprocess and rendering it as a monomodal set. However, this is feasible only if all the initial datasets have the same property type, so that all the voxels of the fusioned model share the same type. Color Fusion (CF) may also be implemented as a post-process of *n* monomodal classification and shading pipelines. However, the flexibility of changing the fusion parameters would not be achieved and the other fusion modes would be difficult to implement. This is why we have designed the new system described below.

Per-Sample queries

The common kernel to all rendering methods are the per-sample queries. Here "sample" should be understood in a large sense either as points for ray-casting, voxels for splatting or traversal methods and texels for 3-D texture mapping. In a multimodal study of *n* initial models, these queries are:

- the set of *n* property values $\mathbf{v} = \{v_i,\ \text{i=1..n}\}$ at the sample location, which can be of any type.

$$f_{prop}(p) \quad = \quad v_0, v_1, \ldots, v_n \tag{3}$$

- the set of *n* gradient vectors $\mathbf{g} = \{g_i,\ \text{i=1..n}\}$

$$f_{grad}(p) \quad = \quad g_0, g_1, \ldots, g_n \tag{4}$$

- the set of *n* materials represented at the sample $\mathbf{m} = \{m_i,\ \text{i=1..n}\}$. These materials can also be of different type (surface, emission and absorption, etc)

$$f_{clas}(p, v) \quad = \quad m_0, m_1, \ldots, m_n \tag{5}$$

These three queries share the problem of data type diversity. The query or computation of each of their output parameters is done on a separate dataset and it can therefore be different. In addition, the property and material queries output parameters may also be of different types. This diversified access to values can be solved using static or dynamic verification. The former solution may result in inefficient case evaluations throughout the algorithms. At the opposite, static verification may be avoided by generating as many code as case combinations exist, but this may be tedious to program and difficult to maintain. Another possibility is to use automatic code generators. However, this would reduce the flexibility to change models while running the application. Dynamic verification and the use of the inheritance and overloading mechanisms provided by object-oriented design warrant a correct access to the data while avoiding unnecessary computations and code repetition. Therefore, in the proposed design, these query functions are methods of the different models that compose a multimodal study.

Shading

Sample shading functions compute a color by applying a shading model to a sample given a gradient vector and a material type. In the current implementation, there are twenty-nine different shading functions which correspond to the combinations of sample type (point, voxel or texel), material type, shading model, depth cueing flag and light sources attenuation flag. For each sample, the appropriate function should be called, without having to traverse the cases tree. Therefore, we have designed a generic shading function class with its corresponding specific implementations. Before rendering, whenever the user changes parameters, the concrete shading function to be applied is installed dynamically. This solution provides the desired flexibility and it is efficient because it avoids per-sample checking. The generic shading function is:

$$\begin{aligned} SampleShading(&sample, gradient, material,\\ &viewingvector, lightmodel, \quad \rightarrow \quad color\\ &depthcueingscale) \end{aligned}$$

Fusion and Decision

The fusion procedures, called at sample level, perform the five pipelines described in section 2. They have also been designed as a generic class whose five specific implementations are instantiated dynamically previous to rendering.

$$\begin{aligned} Fusion(&nvalues, properties,\\ &gradients, materials, \quad \rightarrow \quad property, gradient\\ &colors) \qquad\quad material, color \end{aligned}$$

Inside each fusion pipeline, a decision function is called that either selects (OPP) or mixes (MPP) elements. Similarly to shading and fusion, decision functions may have various input parameters (properties, gradients, material or colors) and several possible implementations depending on user criteria. Therefore, a decision

function class has been designed that includes two categories of functions OPP-based and MPP-based. These functions are also installed dynamically on user request.

The specific OPP and MPP functions can be designed ad-hoc according to a convenient criterion. As an example the *maxproperty* function, which can applied in the first method described in Section 2 for multimodal sets with the same property value type, simply selects the maximum property value. Ad-hoc functions can be fix. Users selects them through the interface according to a key-name. In addition, they can be generated on line from user defined expressions. This case is more flexible but less efficient because it suffers from the processes communication cost overhead. A more general decision function definition consists of specifying output indexes or weight values depending on a combination of input parameter ranges. Let r_k^n a a combination of n value ranges $r^k = (r_1^k, r_2^k, ...r_n^k)$ with k varying between 1 and *nranges*, the number of range value combination. In an OPP scheme, each range combination r_k has associated the chosen element p_{i_k}. In an MPP scheme, each range combination r_k has associated a set of weights $w_{l_k}, l = 1...n$. Then, the decision functions, f_{opp} and f_{mpp} can be defined as:

$$f_{opp}(p_1,...,p_n) = p_{i_k} \quad \exists r_k, k = (1 \ldots nranges) \quad such \quad \forall j : 1 \leq j \leq n : p_j \in r_j^k \quad (6)$$

$$f_{mpp}(p_1,...,p_n) = (w_{l_k},\ldots,w_{l_n}) \quad n = (1 \ldots n) \quad \exists r_k, k = (1 \ldots nranges) \quad such \quad \forall j : 1 \leq j \leq n : p_j \in r_j^k \quad (7)$$

The implementation of these functions using look-up tables, which would be the fastest, is not always feasible because of its huge memory requirements. For a simple multimodal study of two properties with RGBα bytes color fusion, the decision function has an occupancy of 256^8 for an OPP function and $256^8 * 8$ for an MPP function. It addition, user specification of these types of functions may be very tedious. An alternative solution is to define and store a default output index or output weights set plus only a set of range combinations. At run-time, a search within these ranges combination is performed. This is obviously less efficient than direct indexing to look-up tables but it requires less memory and it is easy to specify.

Rendering

The different rendering algorithms are classified into three groups depending on the type of scene traversal that they perform: unsorted object-order (z-buffer methods), sorted object-order (splatting, BTF and FTB traversal, texture mapping) and ray-order (ray-casting). A hybrid traversal mode mixing image and object order is currently under study. In each case, the corresponding database traversal is performed and, when the sample level is reached, the queries and computation exposed above are applied. We enclose below a simplified pseudo-code example of sorted BTF traversal algorithm based on this design.

```
projection
  for (i = min[0]; i! = max[0]; i+ = incr[0]) do
    for (j = min[1]; j! = max[1]; j+ = incr[1]) do
      for (k = min[2]; k! = max[2]; k+ = incr[2]) do
        s = SampleVoxel(ijk)
        (ffusion)(s, cluster, prefgroup, camera, lights,
                  &color)
        if (!BlackColRGBA(color))
                  (splat)(color, p, prefsplat, buffer);
        endif
      endfor
    endfor
  endfor
endproj
```

A material fusion function can be written as:

```
fusion
  n = ObtNObjVisCluster(cluster);
  objvis = ObtFirstObjVisCluster(cluster);
  i = 0;
  while (!LastObjVisCluster(cluster, objvis)) do
        (fprop)(s, objvis, &values[i]);
        (fgrad)(s, objvis, &grad[i]);
        (fmat)(s, &mat[i]);
        objvis = ObtNextObjVisCluster(cluster, objvis);
  endwhile
  (fdecision)(n, values, grad, mat, &outputelem);
  (fshade)(s, ouputelem, lights, camera, &color);
endfus
```

General structure

Figure 1 sketches the internal classes structure of the system. Blue arrows represent relationships while inheritance links. The main class of the system is the rendering database (*bdgvis*). It is a set of multimodal studies called *clusters*. Each cluster is composed of monomodal aligned visualization objects *objvis*. Finally, an *objvis* contains a geometrical object which can be a parametric model (spheres, cylinders, etc.), a boundary representation or a voxel model with different property types. Each level of this hierarchy (bdgvis, cluster and objvis) has associated a camera, a lighting model and rendering preferences (*prefvisgroup*, i.e. group preferences for the database and the clusters and *prefvisobj*, i.e. object preferences for the visualization objects). These preferences consist of general rendering parameters *prefvisgen*, rendering algorithms preferences *prefvisalgor* and objects preferences *prefvisobj*. Examples of general preferences are the depth cueing flag, the light sources attenuation flag, and the 2D and 3D texturing flags. The algorithm preferences are specific to the rendering method being applied, as for instance constant or adaptive sampling for ray-casting or type of footprint for splatting methods. Finally, examples of object preferences are the resolution at which a voxel model should be processed and the type of gradient vector calculation. Having camera, lights and preferences at these three levels opens the possibility to apply different parameters for each object or one common to the cluster or to the database. Material, shading, fusion and decision classes are also shown.

5 Implementation and results

The proposed design has been implemented and tested on a Sun Ultra 60 360MHz. The resulting software correctly handles the different fusion methods with various combinations of input datasets. It currently supports ray-casting, splatting and 3-D texture-mapping. Much effort should still be put on the development of the user interface which is, by now, simple and not very friendly for non computer scientist users.

Color Plates 1 to 5 show examples of use of the system. They represent different fusion methods applied on the same datasets. The simulations have been realized on a 190x220x178 multimodal study composed of 2 bytes-intensity MR images, 1 byte-intensity labeled images and 1 byte per channel (RGB) SPECT images. The labeled model represents the segmented anatomical regions of the brain. In all the simulations the MR materials are hybrid (surface+volume), the labeled materials are surface-only and SPECT

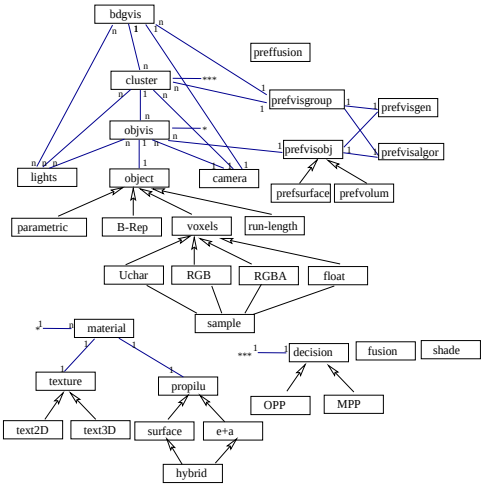

Figure 1: System design

materials are emission+absorption. The shading model applied is Phong + Emission + absorption with light sources atenuation and depth cueing.

Color Plate 1 is an example of property fusion applying OPP mode (method 1). MR values with high intensity SPECT values. Color Plate 2 shows how P&G fusion between labeled data and MR data. It is an OPP case (method 2). The decision function that has been applied selects labeled values with a significant gradient or MR data. Therefore, the image shows MR values everywhere but in the cerebral cortex in which the label is shown. In is red on the right and green on the left. Color Plate 3 is an example of material fusion (MF) applying weighted averages (MPP mode, method 6). It uses the three datasets. The transfer function shows SPECT everywhere except surface MR values indicated by the labeled model. Color Plate 4 shows an use of method 7. A weighted shading of SPECT and MR data has been applied. Color Plate 5 is an example of color blending of shaded MR and SPECT data. It corresponds to method 9.

Color Plate 1 to 3 images have been obtained with ray-casting and Color Plates 4 to 5 with splatting. However, we have proved both algorithms in all the simulation cases. The differences between the two approaches are not very significant neither in image quality nor in computational cost, because the voxel/pixel ratio are low (around 4).

The computational times depend more on the datatypes being merged than on the fusion method. Specifically, the lower times, that range from 55 seconds to 65 seconds, correspond to merging the labeled and the MRI models (1+1 byte) as done for Color plate 2 and 3. Color plates 1 and 5 that merge MRI and SPECT (1 byte +3 bytes) require times of 185 and 165 seconds respectively. Finally, Color plate 4, that uses the three datasets (1byte+1 byte+3bytes) requires 215 seconds.

the five images vary between 125 and 148 seconds. The color fusion simulation (CF) has also be proved using 3D-texture mapping on a SGI Octane with 4MB of texture memory requiring a CPU time of 3.7 seconds in front of 1.2 seconds for a monomodal study, due to the memory swap overhead.

In order to have an estimation of the efficiency of the design, we have implemented a program that realizes the particular case of simulation of the first color plate. The program assumes the given dataset type and fusion method so it does not realize any checking. Next, we have introduced in the code all the verifications that would be necessary in the general case. The computational cost of the simulation with our design is 185.92 seconds. In the simulation without checking it is 184.26 and when all the dynamic verifications ar done, the CPU time is 225.44 seconds. These results show that the system efficiency is close to the specific implementation case and lower than the dynamic verification one. It therefore provides the needed flexibility without loss of efficiency and it avoids code redundancy.

6 Conclusions

In this paper, the direct rendering of multimodal aligned volume models has been addressed. First, different techniques for mixing datasets in the rendering process have been proposed, that may either select one property per point or perform a weighted averaging of them. These techniques also differ in the type of parameters that are used in the fusion (property values, gradients, materials or colors) and in the step of the rendering pipeline at which the fusion is realized.

The requirements of a multimodal rendering system able to provide these different techniques have been analyzed. Next, a design of such a system have been proposed. Finally, an implementation based on this design has been described and some simulations results have been shown.

The next step of our research is the use of the system in order to evaluate in a systematic way how the different fusion methods can be used to outline a specific relationship. This depends on the input datasets, their classification and the shading model applied. We will continue working on medical images.

In addition, we plan to extend our system to other rendering algorithms, such as run-length encoding. According to [H-Pfister et al. 2001], finding good transfer functions is one of the top ten problems in visualization. Finding good decision functions for fusion is not easier. The development of intuitive user interfaces that may help users to define them is therefore an open problem. Automatic decision functions should also be investigated. Another important problem that should be studied is rendering non-aligned datasets. Finally, our system handles scenes in which each object may have its own camera and light sources. An interesting feature to be added would be the composition of different rendering schemes, one per object in a single process.

Acknowledgments: This work has been funded by the project TIC 99-1230-C02-02 from the Ministerio de Educación y Ciencia.

References

AVILA, R., HE, T., HONG, L., KAUFMAN, A., PFISTER, H., SILVA, C., SOBIERAJSKI, L., AND WANG, S. 1994. Volvis: A diversified volume visualization system. *Volume Visualization 1994* (October).

CAI, W., AND SAKAS, G. 1999. Data intermixing and multivolume rendering. *Computer Graphics Forum 18*, 3, 359–368.

ECKEL, G. 1999. Opengl volumizer programmer's guide. *Document n. 0073720001*.

EL-KHALILI, N., BRODLIE, K., AND CRUM, W. 1996. Visualization of medical data by volume rendering of ct and spect images. *UK-EG Conference*, 283–295.

FERRE, M., AND TOST, D. 2001. Integration of multimodal data based on surface registration. *Proceedings of METMBS'01* (June), 73–77.

FERRE, M., PUIG, A., AND TOST, D. 2002. Rendering techniques for multimodal data. *Report LSI- 64-R*.

H-PFISTER, LORENSEN, B., BAJA, C., KINDLMANN, G., SHROEDER, W., AVILA, L., RAGHU, K., MACHIRAJU, R., AND LEE, J. 2001. The transfer function bake-off. *IEEE Computer Graphics and Applications 1*, 4.

HAWKES, D., HILL, D., LEHMANN, E., ROBINSON, G., MAISEY, M., AND COLCHESTER, A. 1990. Preliminary work on the interpretation of spect images wth the aid of registered mr images and a mr derived 3d neuro-anatomical atlas. *3-D Imaging in Medicine*, 241–252.

HU, X., TAN, K., LEVIN, D., PELIZZARI, C., AND CHEN, C. 1990. A volumetric rendering technique for integrated 3-d display of mr and pet data. *3D Imaging in Medicine*, 379–398.

LACROUTE, P. 1995. Fast volume rendering using a shear-warp factorization of the viewing transformation. Tech. rep., Departments of Electrical Engineeering and Computer Science, Stanford University, Setember.

LEMOINE, D., BARILLOT, C., GIBAUD, B., AND PASQUALINI, E. 1991. An anatomical-based 3d registration of multimodality and atlas data in neurosurgery. *Information Processing in Medical Imaging*, 154–164.

PELIZZARI, C., CHEN, G., SPELBRING, D., WEICHSELBAUM, R., AND CHEN, C. 1989. Accurate three-dimensional registration of ct, pet, and/or mr images of the brain. *Journal of Computer-Assisted Tomography 13*, 1 (January), 20–26.

SCHROEDER, W., MARTIN, K., AND LORENSEN, B. 1998. *The Visualization Toolkit*. Prentice hall.

SPETSIERIS, P., DHAWAN, V., ISHIKAWA, T., AND EIDELBERG, D. 1995. Interactive visualization of coregistered tomographic images. *BioMedVis'95*, 58–63.

VAN DER ELSEN, P. 1993. *Multimodality Matching of Brain Images*. PhD thesis, Utrecht University.

ZUIDERVELD, K., KONING, A., STOKKING, R., MAINTZ, J., APPELMAN, F., AND VIERGEVER, M. 1996. Multimodality visualization of medical volume data. *Computer and Graphics 20*, 6, 775–791.

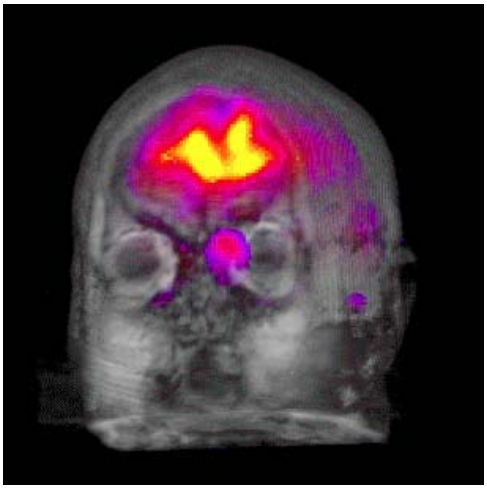

Figure 2: PF example.

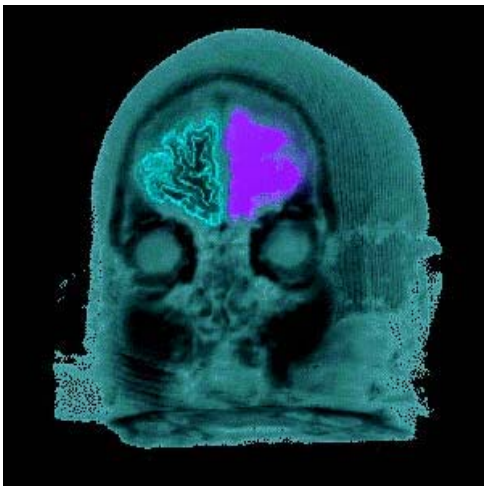

Figure 3: P&GF example.

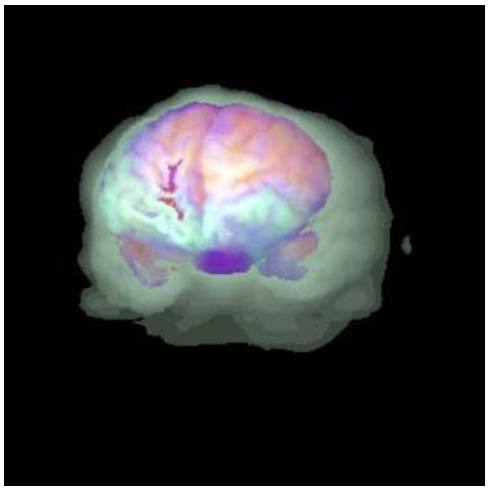

Figure 4: MF example.

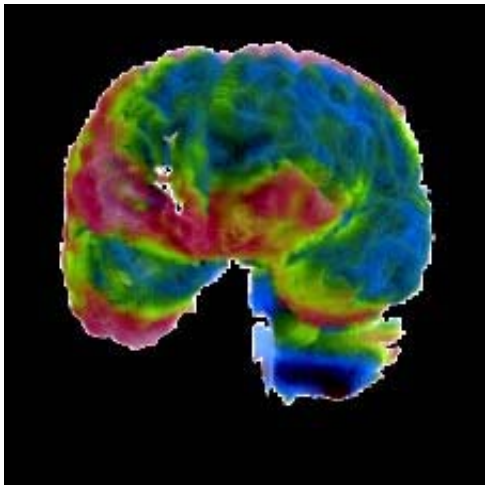

Figure 5: SF example.

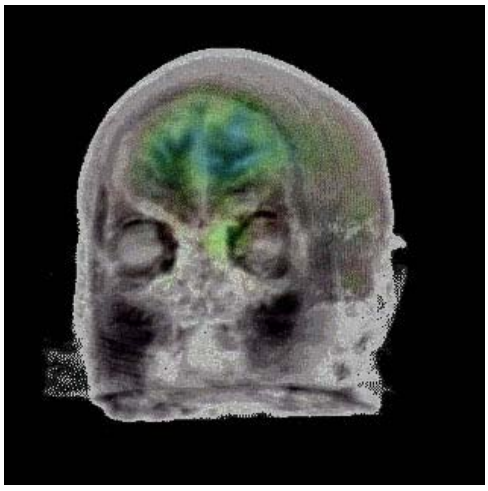

Figure 6: CF example.