Volumes of Expression
Artistic Modelling and Rendering of Volume Datasets

S. M. F. Treavett, M. Chen, R. Satherley and M. W. Jones
Department of Computer Science, University of Wales Swansea, United Kingdom
{cssteve, m.chen, csrich, m.w.jones}@swansea.ac.uk

Abstract

This paper presents the design and implementation of artistic effects in modelling and rendering of volume datasets. Following different stages of a volume-based graphics pipeline, we examine various properties of volume data, and illustrate how expressive and non-photorealistic effects can be implemented. We demonstrate that the true 3D nature of volume data makes it particularly applicable for this use, allowing the addition of complex effects at the modelling stage as well as during rendering.

1. Introduction

The advances in computer graphics have successfully produced a huge collection of techniques for synthesising photorealistic images. In particular, these techniques have become essential tools in areas such as computer-aided design, computer games, virtual environments, scientific visualisation and medical imaging. In recent years, the surge of applications in the entertainment and art world directed a substantial amount of research efforts towards expressive modelling and non-photorealistic (NPR) rendering. The most representative work includes: Strassman's painstaking model of a real life brush [1], Haeberli's and Mier's oil paintings [2, 3], Curtis' watercolour imitations [4], Sousa and Buchanan's work on pencil drawings [5, 6] and a variety of methods for generating pen-and-ink style images [7, 8, 9, 10]. Almost all these techniques were designed for surface-based graphics objects, and were integrated with traditional surface-based graphics pipelines.

The past decade has witnessed significant advances in volume visualisation [11, 12, 13], driven mainly by applications such as medical imaging and scientific computation. The work in this field has produced a number of volume rendering methods that enable 3D information in a volume dataset to be selectively rendered into 2D images. A volume dataset facilitates a true 3D representation of the interior, as well as the exterior, of an object. Volume datasets are commonly found in a multi-valued and multivariate form (e.g. the visible human dataset [14]), where the geometrical and physical properties (such as colour, opacity, density, etc.) of an object are defined upon a point set in a consistent manner. Hence a volume representation often contains more information than a surface representation, and can be used to generate potentially more interesting artistic effects. With modern computer hardware, volume-based modelling and rendering can easily be performed on an ordinary workstation, and is beginning to evolve into a general purpose graphics technique [15, 16]. Figure 1 typifies such an evolution with two images: (a) an image generated as a visualisation of an MRI (magnetic resonance imaging) dataset, (b) an artistic image synthesised from a graphics scene composed of multiple volume objects, some of which were built from a CT (computed tomography) dataset. Both synthesised using volume datasets but represent entirely different objectives.

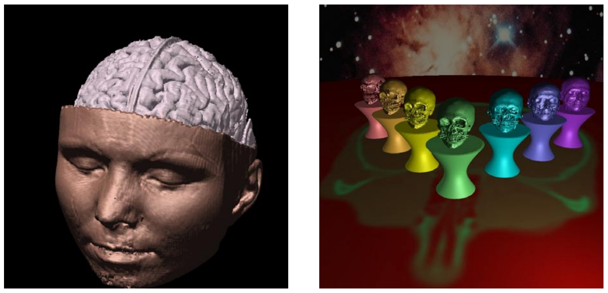
(a) (b)

Figure 1. From visualisation to graphics.

Some efforts have been made to introduce NPR techniques in aid of volume visualisation [17, 18, 19, 20]. However, more serious efforts are necessary for integrating various artistic effects systematically into a software framework for volume-based techniques. It is such a need that motivated our investigation into artistic modelling and rendering of volume representations.

This paper presents a study on a volume-based graphics pipeline with special reference to the place of expressive and NPR effects within its structure. Each component of the pipeline will be examined in turn, and discussions made of how artistic effects can be introduced at that stage. In particular, we focus on several modelling and rendering techniques that are important in the context of volume datasets. We use the term *filter* as an abstract notion to discuss various algorithms involved in the artistic modelling and rendering of volume datasets. Throughout the paper, we consciously confine the use of the adjective *non-photorealistic (NPR)* to the simulation of various hand painting effects, such as pen-and-ink drawing, and employ the adjective *expressive* to imply surreal features, such as object and viewing deformation, which may be rendered using either photorealistic or non-photorealistic methods. These effects are collectively referred to as *artistic* effects.

2. A Volume-Based Pipeline for Synthesising Artistic Images

In this section we will discuss the general structure of a volume-based graphics pipeline, as shown in Figure 2, where dark boxes highlight operational components that incorporate artistic modelling and rendering features. At the higher level, it is similar in form to its surface based analogue. In essence it is broken down into three main stages, namely *modelling space*, *rendering space*, and *image space*. Each of these stages facilitates the generation of different classes of effects. This is determined by the amount and type of information available at each stage as well as the operations performed therein. The characteristics of each stage of the pipeline are described in the rest of this section, while the methods and algorithms operated at each stage will be discussed in Sections 3, 4 and 5 respectively.

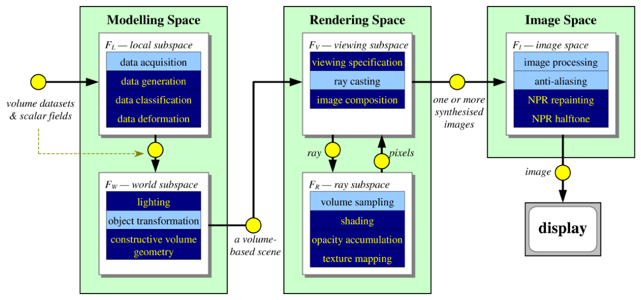

Figure 2. A volume-based graphics pipeline.

2.1. Modelling Space

A general volume representation V consists of a point set P, and a d-dimensional vector of values $\{x_1, x_2, ..., x_d\}$ defined at each point $p \in P$, and a set of interpolation functions $I_1, I_2, ..., I_d$. Each interpolation function I_i is specified upon all values $X_i = \{x_i\}$ in the *ith* dimension, and it defines a value at every point in the volume. In other words, I_i defines a scalar field, and V is thus a discrete specification of d scalar fields. The most commonly-used volume representation is a multi-valued representation that contains a regular grid of voxels, each is associated with a value $x \in \mathbb{R}$ *or* $\mathbb{I}$. The values at non-grid points are usually obtained using tri-linear interpolation of neighbouring voxels. The most significant feature of such a volume model is its homogeneous data organisation that facilitates simple data structures and efficient implementation. Our pipeline was primarily designed and implemented based on a multivariate extension of this representation. It also accepts mathematical and procedural specifications of scalar fields as input.

Modelling tasks are typically conducted in the *local subspace* for each volume as well as the *world subspace* for an entire scene. Let us consider the functionality of these subspaces as two filters F_L and F_W in an abstract term. The primary role of F_L is to preprocess each raw dataset, and prepare a volume object V that comprises necessary object attributes to be used later in the rendering space. These attributes may include: X_{geo}, $X_{opacity}$, X_{red}, X_{green} and X_{blue}. X_{geo} is usually acquired as raw data, and is used to define iso-surfaces, normals and mappings in data classification. In addition, there are also attributes associated to the whole volume object, such as texture identifier and parameters. We use O_{att} to denote these attributes collectively. We thereby consider F_L as a mapping from a raw dataset to an volume object, $O = \langle O_{att}, P, X_{geo}, X_{opacity}, X_{red}, X_{green}, X_{blue}\rangle$. Given a set of volume objects $O_1, O_2, ..., O_n$, F_W is a mapping from these objects to a scene, $S = \langle S_{att}, O'_1, O'_2, ..., O'_n\rangle$, where S_{att} denotes a collection of scenery attributes such as light sources.

Any artistic effect which is introduced at the modelling stage will normally remain whatever other effects are introduced later, and will maintain its viewing consistency and coherency regardless of the viewing system.

2.2. Rendering Space

The concept of rendering space used in this paper encompasses two aspects of direct volume rendering, namely the *viewing subspace* and the *ray subspace*. Operations in these two subspaces are intrinsically related in terms of the interaction between them and the information which they share. We use VW_{att} to denote the specification of view plane and camera attributes. In addition, the viewing subspace also contains one or more image buffers, $IB_1, IB_2, ..., IB_k$. Later we will see these buffers may be used for some unconventional ‘images’. The generation of these 'images' thus becomes the primary goal of the operations in the viewing subspace. These operations are collectively defined as a filter F_V.

From the camera, rays are cast into the scene (or the world subspace). As we treat each ray as an independent entity, we define a single ray casting as a filter F_R. F_V and F_R are not operated in a sequential manner, but much in a master-slave fashion.

Note that with NPR rendering, rays may not correspond to the pixels in image buffers in either number or order. It is also common that each ray, F_R may update more than one pixel in each image buffer. Unlike the techniques associated with the modelling space, artistic effects applied here do not change the scene permanently. They are repeatable by using the same rendering techniques again but their only legacy is the output images that they produce. The vast amount of information available at this stage means that a great number of effects can be produced. For this reason, the rendering space in a surface-based graphics pipeline has been a focus in NPR (e.g. [3, 21]). One factor worth noting is that, as the techniques are mostly implemented in F_R, the generated artistic effects are often view-dependent. This means that great care must be taken in generating coherent frames for an animation, and indeed, a substantial number of previous NPR implementation in the rendering space do generate undesirable artefacts when animated [22].

2.3. Image Space

Image space, as defined in this paper, encompasses all techniques which manipulate 2D images, including those for image processing [23]. This was the first and, to date, most extensively exploited domain for generating NPR effects despite its lack of spatial information [2, 8, 24]. This is partially due to the availability of real-life images, and partially due to the difficulties in tempering any graphics pipeline provided in a commercial package. Certain techniques successfully extracted from a graphics pipeline additional 3D information associated with a synthesised image. With such information, they demonstrated an extra degree of accuracy and capability in producing NPR effects.

Our pipeline is designed to forward necessary 3D information, such as depth and normals, to the image space through multiple image buffers, $\{IB_1, IB_2, ..., IB_k\}$. In most cases, the generation of such 3D information is a necessity in the rendering space, and takes little extra computing time. As only 2D buffers are used to store the information, the extra storage requirement is relatively insignificant in comparison with storage space for volume datasets. Given a set of image buffers, the operations in the image space are collectively defined by filter F_I as a mapping from these buffers to an final resultant image IM.

2.4. Parameter Control

The homogenous nature of data flowing in the above pipeline can be exploited in many ways. Firstly images can be treated as a special case of volume datasets. Secondly, many effects can be defined and implemented as a volume dataset or a scalar field. Our pipeline is completed with a large number of expressive and NPR filters, many of which are stored in the form of *control volumes*. All control volumes have a similar access interface, though each may be stored as a volume dataset, a mathematical expression, or a procedure. As illustrated in Figure 3, when various data flows through the pipeline, parameters for expressive and NPR effects trigger the interaction between the data and appropriate control volumes.

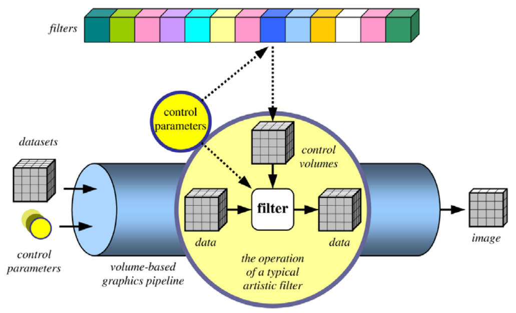

Figure 3. A control volume in action.

The possible ways of using such a control volume are numerous. It may simply contain a scaling value for every voxel in a volume object to change the shape of isosurfaces or, in a different situation, may provide a complicated mathematical mapping that defines 3D NPR strokes. Figures 4(a-c) show three examples of using control volumes to (a) perform distortion, (b) modify opacity, and (c) apply a mathematical solid texture.

3. Artistic Modelling

In volume graphics there are two general approaches for the introduction of expressive effects into the model, namely, by manipulating existing features and by introducing NPR features directly.

3.1. Artistic Manipulation

Consider a volume object O as defined in Section 2.1. It is possible to introduce expressive effects by altering the attributes of O. Perhaps the most powerful branch of these effects can be produced by simply manipulating X_{geo}, which determines the iso-surfaces in O and normal estimation during the rendering. In addition, any alteration prior to the

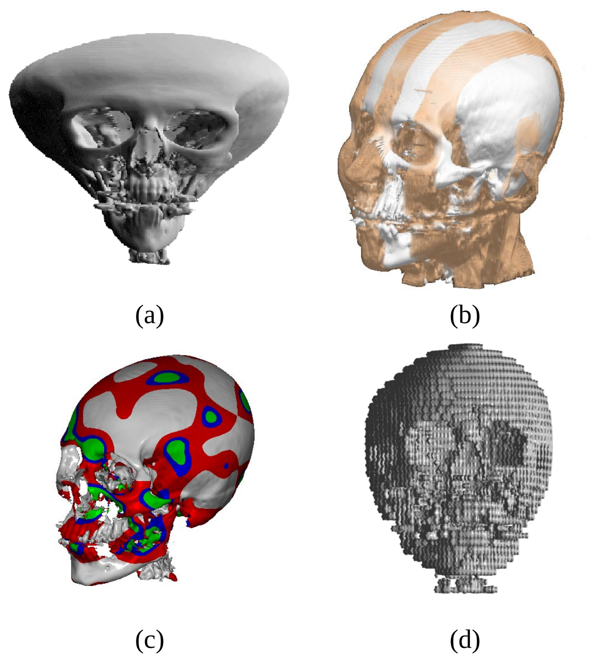

Figure 4. Examples of artistic manipulation.

data classification stage (Figure 2) may also influence other fields if they are absent in the raw datasets. The *control volume* is an effective mechanism for implementing filters for such manipulation. Modifications can also be made to $X_{opacity}$, X_{red}, X_{green} and X_{blue} in a similar manner. Our pipeline is currently equipped with the following set of filters:

- $f_{distort}(X, C) \Rightarrow X'$, where X and X' are volume datasets or scalar fields and C is a 3-variate control volume for controlling the translation in x, y and z directions respectively. The filter assigns each voxel $x' \in X'$ with a scalar value in X at the position which is a translation of the corresponding $x \in X$ by $(cx, cy, cz) \in C$. Figure 4(a) was generated using such a filter, and shows a distinct change to the geometry of the model. The control volumes for $f_{distort}$ are typically computed using various volume distortion algorithms [25].

- $f_{scale}(X, C) \Rightarrow X'$, where C is a single-variate control volume. The filter essentially multiplies each voxel $x \in X$ by the corresponding scalar value defined by C. Figure 4(b) shows the application of such a filter to the $X_{opacity}$ field of the volume object.

- $f_{stroke}(X, C_{att}) \Rightarrow X'$, where C_{att} is a set control parameters, for specifying stroke type, size, density, etc. The filter samples voxels in X randomly or according to the specified density, and it replaces each sampled x with 'stroke' in a subdomain in X' corresponding to x. Figure 4(d) gives a simple example where spherical strokes are applied to a volume object by this filter.

- $f_{lookup}(X, C_{LUT}) \Rightarrow X'$, where C_{LUT} is a look-up table. This filter is a standard component in most volume-based graphics pipelines for data classification. They can also be utilised to produce expressive effects. For instance, the X_{red} field of the model in Figure 4(c) was defined as $f_{lookup}(X_{noise}, C_{LUT}) \Rightarrow X_{red}$, and X_{green} and X_{blue} were defined in a similar manner.

One major strength of a volume representation is the inherent presence of 3D opacity and colour information in the model. Thus artistic effects that may only be introduced as textures in a surface-based pipeline can now be actually modelled. These alterations will therefore not show any of the commonly seen anomalies, such as the *shower door* effect, in animation.

3.2. Artistic Additions

With artistic additions, expressive features are created and introduced directly into a volume object. Here we focus on physical objects such as strokes and masks which are incorporated into an existing volume object to form a composite object. Those effects, which are only defined as additional attributes in O_{att} at this stage, but are realised in later stages of the pipeline, will be considered in later sections.

One tool which has been found highly useful for the introduction of these effects is constructive volume geometry (CVG) [26], which is a major generalisation of constructive surface geometry (CSG). CVG is designed to work with volume datasets and mathematically defined scalar fields, and it operates in the real domain $\mathbb{R}$.

Some interesting effects, such as those shown in Figure 5 can be generated by combining two or more volumes together using CVG. In Figure 5(a) three volumes are used: a CT dataset and two 3D stroke volumes coloured differently. The distribution of the 3D strokes can be controlled in a number of ways. In this case, *detractors* and *attractors*, which are specified as invisible light sources, are used to control the position and density of the strokes. The CVG operation used to generate this image is $O = (O_G \cup O_P) \cap O_{head}$ where O_G and O_P are two stroke volumes, and O_{head} is a CT dataset. A reference image is shown in the corner of Figure 5(a), which was generated by the CVG operation $O' = O \cup O_{skull}$.

A further example showing the introduction of *paint splats* is given in Figure 5(b). Once again the splats are directly added to the model thus creating an effect which will generate totally coherent animation sequences. One medium of traditional art which can be replicated easily

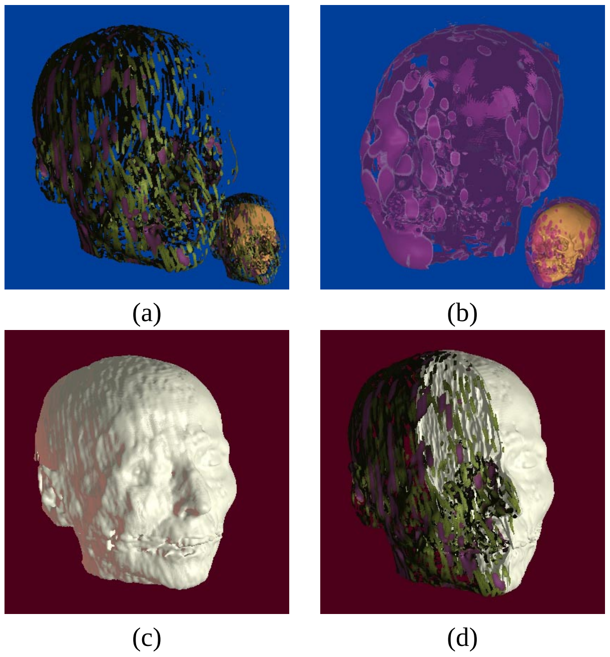
(a) (b) (c) (d)

Figure 5. Artistic additions using CVG.

using CVG is sculpture. Figure 5(c) shows the CT head that has the stroke volume O_G subtracted from it, that is, $O = O_{head} - O_G$. As the stokes are concentrated around the surface of the head, this gives the effect of a rough carving technique. In our pipeline, the CVG method can be viewed in abstract as a filter $f_{CVG}(op, O_a, O_b) \Rightarrow O_c$, which can be applied to volume objects recursively. Figure 5(d) shows the application of a more complex CVG term.

4. Artistic Rendering

There are two fundamental approaches to the direct rendering of volume objects, namely *ray casting* and *forward projection*. The former is further classified into *direct volume rendering* [12] which treats a volume as an amorphous object, and *direct surface rendering* [27] which is interested in one or more iso-surfaces in the volume. The latter is represented by work on projecting voxels as solid 2D or 3D objects in the early days, lately as amorphous objects as in splatting [13]. The rendering stage in our pipeline is based on the ray casting mechanism, and it supports both direct volume rendering and direct surface rendering. In addition, rays can also be fired from randomly selected pixels in the image plane, and this is particularly useful in producing some NPR effects.

It is at the rendering stage that we have the richest concentration of information that can be used for generation of various expressive and NPR effects. From the definition of the viewing system, through to the final rendering of the colour at each pixel in a synthesised image we have opportunities to introduce artistic effects by manipulating the viewing system, normals, opacity, colour, geometry, iso-surface, textures and hypertextures.

4.1. Viewing System

As traditional computer graphics focuses on the synthesis of photorealistic images, the viewing system is normally designed to simulate a conventional camera in a reasonably accurate manner. In such a viewing system, a flat image plane represents the film in a camera, while rays fired from a focal point into the scene follow the reverse paths of real light rays. Any manipulation of this system will lead to anomalies in the synthesised image. By careful control of these manipulations these anomalies can be directed so that they produce pleasing expressive effects. Work has been previously published on using fields to bend the ray paths [28] and produce interesting effects in images. We have incorporated a similar feature in our pipeline as a filter for manipulating primary rays, $f_{ray}(R, VW_{att}) \Rightarrow R'$, where R and R' are data structures of rays, and VW_{att} contains all attributes of a viewing system.

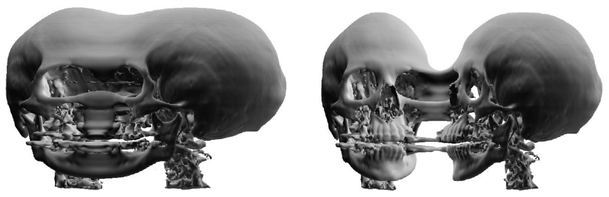

Figure 6. ‘Meeting of minds’.

The filter transforms a flat image plane into a curved plane, and warps the direction of each ray according to the new image plane before firing it into the scene. The shape of the image plane is defined by a Catmull-Rom spline and can take almost any shape with appropriate control parameters. The link between the warped rays with the original image plane is maintained, and the synthesised pixel values can then be mapped back onto a flat image giving extremely interesting visual effects. The images shown in Figure 6 give the impression of two skulls merging where there is in fact only one. The image on the right uses a tighter curve than that on the left.

4.2. Normals

During ray casting (i.e. in the ray subspace), each ray is sampled at regular intervals in searching for an intersection with a specific iso-surface (in direct surface rendering) or non-transparent matter (in direct volume rendering). When a ray hits the specific iso-surface or a non-transparent sampling point, a normal is estimated, usually using the central difference method. Manipulation of this data has been used to generate a variety of effects in surface based graphics pipelines. Bump mapping is perhaps the most well known example of such effects. In our pipeline, a normal N can be manipulated in a number of ways, including the use of filters defined in the previous sections. For instance, $f_{distort}$ can be applied to each dimensional element of a normal at the sampling point in relation to its position in the local volume subspace.

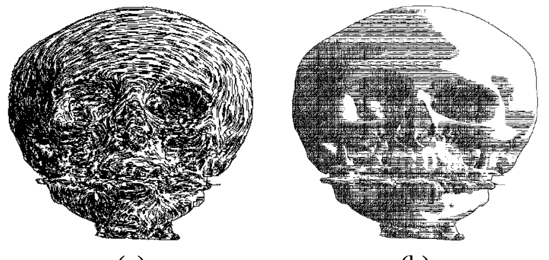

Figure 7. Painting with normals.

In addition, a special filter $f_{npr\text{-}shade}(R, t, N, C_{att})$ is used to shade a sampling point with NPR effects, where R is a ray, t is a sampling point, N is the sampled normal at t, and C_{att} is a set of control parameters. We can easily observe the difference between $f_{npr\text{-}shade}$ and a traditional shading filter where the normal would be used to translate the sampled colour into a shaded colour. The $f_{npr\text{-}shade}$ filter determines a shaded colour without the knowledge of the sampled colour. Moreover, it normally applies colour to a number of pixels in the image plane, facilitating the painting of a stroke.

In Figure 7(a) the normals themselves are drawn as strokes onto the image, after being rotated according to the corresponding ray direction. This gives the impression of a coarsely drawn pencil sketch. Similar to a photorealistic illumination model, the size and density of NPR strokes can also be influenced by light sources. Figure 7(b) shows an image generated using a prioritised hashing texture [7], where the priority is determined by an illumination model.

4.3. Colour and Opacity

We have already discussed the introduction of artistic effects through the modification of opacity and colour fields in the modelling stage. Many such modifications can also be implemented in the rendering stage using filters, though it is usually more expensive in terms of computation. The major difference is that these filters are normally implemented in the general form of $f(R, t, x, C) \Rightarrow x'$ or $f(R, t, x, C_{att}) \Rightarrow x'$ instead of $f(X, C) \Rightarrow X'$, where x and x' are scalar values, while X and X' are fields. The main advantage of manipulating colour and opacity at this stage is the possibility of producing effects independent from the resolution of volume datasets, and those dependent on rendering parameters such as z-depth, ray direction and light sources.

4.4. Geometry

The position, size and shape of objects can be used evocatively to add "expressions" in a graphics scene. Often the human artist may present things out of context or proportion for some kind of emphasis. Traditional volume rendering requires that we make the majority of these types of classifications at modelling time, however in our volume graphics pipeline, a number of these decisions can be deferred to rendering space.

For example we can extract from a 3D scene the distance of objects from the viewpoint. This can be used to overcome the complex problem encountered in 2D image analysis of which pixels are part of which object. Particularly when overlapping objects are of the same colour or are similarly shadowed. Figure 8 shows an outline image of a plane drawn according to the relative distance information [29].

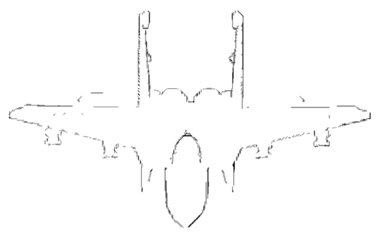

Figure 8. Using distance information.

4.5. Texture Mapping

A powerful technique often made use of in both photorealistic and NPR graphics is texture mapping, which applies desirable surface details to an object by utilising a 2D or 3D texture during the rendering process. As a volume is modelled by either a volume dataset or a scalar field, solid textures [30] work extremely well with a volume object. The concept of *control volume* in our pipeline means that such a texture can be integrated into the rendering stage in a manner consistent with other artistic effects.

Figure 9(a) shows an image rendered with a mathematically defined texture which determines the colour under the control of an illumination model [19]. When a specific iso-surface is detected at t, we apply a composite operation to the sampling point as follows (i) $f_{npr\text{-}shade}(R, t, N, C_{att}) \Rightarrow intensity$, (ii) $C_{att}.density * (1 - intensity) \Rightarrow C_{att}.density$, (iii) $f_{texture}(R, t, C_{att}) \Rightarrow colour$. This particular $f_{npr-shade}$ applies a simple shading algorithm to determine the light intensity at the sampling point. This intensity value is then used to alter the density of the NPR texture to be applied to the point. $f_{texture}$ returns a colour (grey-level in this case) indicating if the sampling point is on the ink-path, which would be evenly distributed around the head if the density remained a constant. Such a composite operation is called a *macro-filter* in our pipeline. Our ray casting algorithm is also capable of combining photo-realistic and NPR rendering as demonstrated by Figure 9(b).

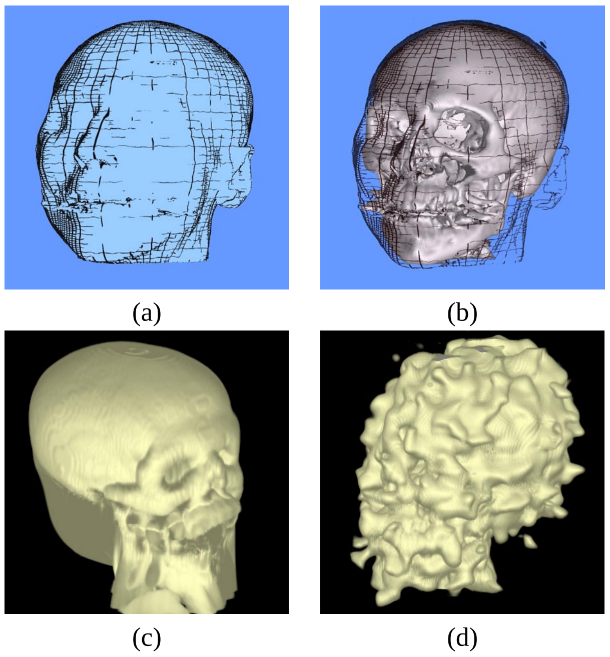

Figure 9. Texture mapping.

4.6. Hypertextures

Traditional textures exist only on the object in question. Using hypertextures [31, 32] an effect can be built that fills some of the space around the object. Such textures are controlled by the density of a surface boundary, which can be obtained by calculating the distance to that surface.

In the case of a volume object, we normally consider a specific iso-surface. The density values are usually stored in a *distance volume* where each voxel value holds the minimum distance from the voxel to the iso-surface. A filter $f_{distvol}(X, \alpha)$ takes a volume X and an iso-value α, and computes a distance volume. Because the iso-surface is usually not mathematically definable, $f_{distvol}$ computes D through a series of *distance transforms* [32]. In general, hypertexture can be viewed as the application of a filter to a distance volume. Such an operation often involves four types of functions, *noise*, *turbulence*, *bias* (which controls the density variation), and *gain* (which controls the rate at which density changes). The general form of the Hypertexture is $f_{hypertexture}(R, t, d, C_{att}) \Rightarrow \langle opacity, colour \rangle$.

Many hypertextures can be generated in this way. The image shown in Figure 9(c) shows a melting hypertexture applied the skull dataset. The epitomal ‘jaw dropping’ experience. Figure 9(d) illustrates the use of low frequency noise applied across the whole skull surface.

5. Artistic Image Composition

There is a rich collection of NPR algorithms for processing photographs and synthesised images in an artistic manner. Many of these algorithms rely almost entirely on the RGB information in an image itself, the resultant images often lack a hint of spatial cue, especially when dealing with images synthesised from 3D scenes. There are a number of such algorithms available at this stage of the pipeline as filters [24], in the form of $f_{image}(IM, C_{att}) \Rightarrow IM'$, however, here we focus on algorithms that are designed to utilise 3D information and produce spatial cue in the final result.

As mentioned briefly in Section 2, not only can the rendering stage of our pipeline produce a synthesised image for the image space, it can also forward spatial information in the form of images. For example, the value of t at which each ray hits an iso-surface can be collectively stored in an IB with real values. Similarly, normals and curvatures can also be forwarded [19]. With this spatial information, the synthesised image can be repainted with some artistic effects using a filter $f_{repaint}(IB_1, IB_2, ..., IB_k, C_{att}) \Rightarrow IM$. Figure 10 shows a pen and ink image rendered in image space (large image) based on a combination of information in several image buffers (small images) that represent distance, normals, shade, vertical curvature and horizontal curvature. This approach is called 2^+D NPR rendering.

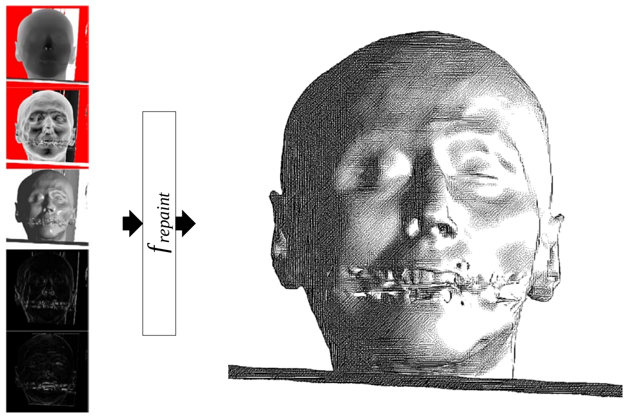

Figure 10. 2^+D Pen and ink painting.

6. Conclusions

In this paper it has been demonstrated that expressive and NPR effects can be introduced at almost every stage of a volume-based graphics pipeline. The effects introduced in the modelling space have the desirable coherency for animation; those realised in the rendering space have access to a variety of information; and those implemented in the image space have shown their capability in producing the profound realism of human drawings.

Significantly, in addition to those effects previously implemented in surface-based pipelines, we have introduced some new effects that utilise information intrinsic to volume representations. Some of these new effects (e.g. opacity manipulation) would present a significant challenge in a surface-based graphics pipeline. Volume representations also make the implementation of these effects easier as most of them are specified by a volume dataset or a scalar field. The concepts of filters and control volumes have shown a degree of consistency in the design and implementation of the pipeline, though they are not always the most efficient in terms of computation and storage.

We believe strongly that there is enormous potential for artistic modelling and rendering with volume representations, especially if the techniques are placed in the hands of an artist. Our future work will focus on a sophisticated mechanism for hierarchical specification of filters, which would facilitate less storage requirement, and potentially faster computation. This will enable us to transfer our research pipeline into a practical software tool.

References

[1] S. Strassman. Hairy brushes. *Proc. SIGGRAPH '86*, 225–232, 1986.

[2] P. Haeberli. Paint by numbers: Abstract image representation. *Proc. SIGGRAPH '90*, 24(4):207–214, 1990.

[3] B. J. Meier and Walt Disney Feature Animation. Painterly rendering for animation. *Proc. SIGGRAPH '96*, 477–485, 1996.

[4] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. Salesin. Computer-generated watercolour. *Proc. SIGGRAPH '97*, 1997.

[5] M. Sousa and J. Buchanan. Computer-generated graphite pencil rendering of polygonal models. *Proc. Eurographics*, 19(3), 1999.

[6] M. Sousa and J. Buchanan. Observational model of blenders and erasers in computer-generated pencil rendering. *Proc. Graphics Interfaces '99*, 1999.

[7] M. Salisbury, S. Anderson, R. Barzel, and D. Salesin. Interactive pen-and-ink illustration. *Proc. SIGGRAPH '94*, 101–108, 1994.

[8] G. Winkenbach and D. Salesin. Computer-generated pen-and-ink illustration. *Proc. SIGGRAPH '94*, 91–100, 1994.

[9] G. Elber. Interactive line art rendering of freeform surfaces. *Proc. Eurographics*, 18(3), 1999.

[10] J. Hamel and T. Strothotte. Capturing and re-using rendition styles for non-photorealistic rendering. *Proc. Eurographics*, 18(3), 1999.

[11] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Proc. SIGGRAPH '87*, 21(4):163–169, 1987.

[12] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[13] L. Westover. Footprint evaluation for volume rendering. *Proc. SIGGRAPH '90*, 24(4):367–376, 1990.

[14] V. Spitzer and M. Ackerman. The visible human male: A technical report. *Journal of the American Medical Informatics Association*, 13(2):118–130, 1996.

[15] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *IEEE Computer*, 26(7):51–64, 1993.

[16] M. Chen, A. Kaufman, and R. Yagel, editors. *Volume Graphics*. Springer, London, 2000.

[17] M. Levoy, H. Fuchs, el al. Volume rendering in radiation treatment planning. *Proc. First Conference on Visualization in Biomedical Computing*, 4–10, IEEE Computer Society Press, Atlanta, Georgia, May, 1990.

[18] D. Ebert and P. Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *Proc. IEEE Visualization, Salt Lake City, Utah*, 195–202, 2000.

[19] S. Treavett and M. Chen. Pen-and-Ink Rendering in Volume Visualisation. *Proc. IEEE Visualization, Salt Lake City, Utah*, 203–210, 2000.

[20] D. Laidlaw (Organiser). Art and visulaization: Oil and water? *Proc. IEEE Visualisation*, 507–509 (Panel), 1998.

[21] P. Hall. Nonphotorealistic rendering by q-mapping. *Computer Graphics Forum*, 18(1):27–39, 1999.

[22] M. Masuch, S. Schlechtweg, and B. Schönwälder. daLi! - Drawing animated lines! *Proc. Simulation und Animation '97*, 87–96, 1997.

[23] A. Low. *Introductory Computer Vision and Image Processing*. McGraw-Hill, 1991.

[24] S. Treavett and M. Chen. Statistical techniques for the automated synthesis of non-photorealistic images. *Proc. 15th Eurographics UK Conference*, 1997.

[25] M. Chen, M. Jones, and P. Townsend. Volume distortion and morphing using disk fields. *Computers and Graphics*, 20(4):567–575, 1996.

[26] M. Chen and J.V. Tucker, Constructive volume geometry. *Computer Graphics Forum*, 19(4):281–293, 2000.

[27] M. Jones. *The Visualisation of Regular Three Dimensional Data*. PhD thesis, University of Wales, Swansea, 1995.

[28] R. Yagel. Classification and survey of algorithms for volume viewing. *SIGGRAPH tutorial notes (course no. 34)*, 1996.

[29] M. Šrámek and A. Kaufman. Object voxelization by filtering. *IEEE Symposium on Volume Visualization*, 111–118, 1998.

[30] D. Peachy. Solid texturing of complex surfaces. *Proc. SIGGRAPH '85*, 19(3):279–286, 1985.

[31] K. Perlin and E. Hoffert. Hypertextures. *Proc. SIGGRAPH '89*, 23(3):253–262, 1989.

[32] R. Satherley and M. Jones. Extending hypertextures to non-geometrically defineable volume data. *Volume Graphics*, M. Chen, A.E. Kaufman and R. Yagel (eds.), 211–225, 2000