Image-Swept Volumes

Andrew S. Winter and Min Chen

Department of Computer Science, University of Wales Swansea, Swansea SA2 8PP, United Kingdom
csandrew@swansea.ac.uk, m.chen@swansea.ac.uk

Abstract

Many graphical objects can be represented by swept volumes (including its subset – generalised cylinders) by sweeping 2D or 3D templates along 3D trajectories. In this paper, we present a new approach for constructing swept volumes using image templates. We utilise scalar fields as our underlying data type, and employ volume ray casting techniques for rendering swept volumes in their original sweeping specifications as well as in their voxelised approximations. In addition to some simple image-swept volumes, we also treat multi-channel image templates, video templates, generalised sweeps, and self-intersecting trajectories. This approach enables us to model swept volumes with heterogeneous interiors and amorphous effects. It also facilitates the use of constructive volume geometry for creating complex scenes in both modelling and rendering space.

Keywords: image-swept volume, volume sweeping, generalised cylinder, volume modelling, volume rendering, constructive volume geometry

1. Introduction

Volume sweeping is a geometric method for defining a spatial domain. It was developed in parallel with many important solid modelling schemes (such as boundary representation, spatial partitioning and constructive solid geometry)[36, 37], and has an indispensable role in graphical modelling, computer-aided design, numerically controlled machining verification and simulation, collision detection and robot analysis[9, 3].

In sweep representations, an object is specified by sweeping a template (typically a closed 2D contour) along a trajectory (typically a parametric curve). A sweep specification may be rendered directly as a solid model, or converted to a boundary representation. Physical attributes, such as opacity and colour, are normally assigned to the object homogeneously or using texture mapping. The existing approach normally requires each template to satisfy a set of geometric conditions such as bivariance and closeness. Despite photographic imagery being a convenient and effective means of capturing a cross-section, images do not satisfy the conditions of "well-defined" geometry.

This paper is concerned with a novel approach for constructing swept volumes using image templates directly. This approach enables us to model swept volumes with hetero-

geneous interiors and amorphous effects. In Section 2, we will give a brief review of the previous research in swept volume and generalised cylinders, and will draw attention to the relevance of volume/field-based modelling and rendering techniques. In Section 3 we will outline a general conceptual framework for defining *image-swept volumes*, and for using them to build *spatial objects*. The modelling of image-swept volumes will be the main focus of Section 4, where we will describe methods for modelling image-swept volumes, and for approximating image-swept volumes via voxelisation. In Section 5, we will first discuss our general approach for rendering spatial objects including voxelised swept volumes, and then describe methods for directly rendering a few classes of image-swept volumes in their original specifications. We will present our observations and concluding remarks in Section 6.

2. Previous Work

The general description of sweep representations, together with other geometric modelling schemes, can be found in a number of surveys and books[36, 37, 21, 20]. Recently, in two separate reviews, Blackmore et al. and Abdel-Malek et al. provided a comprehensive insight into the major developments in the field, and applications of swept volumes[9, 3].

Although simple sweeps are easy to construct and are widely used, generalised sweeps pose various mathematical and computational issues. A substantial amount of research effort has been placed on the calculation of properties associated with swept volumes. Methods have been developed for:

- determining the boundary of an object swept with both its template (2D or 3D) and trajectory defined in parametric forms[10, 1, 32];
- generating multivariate solids using multiple sweeps[24, 4];
- constructing sweeps from implicit surfaces[5], Fourier templates[6] and image templates[47];
- modelling deformable sweeps using operations such as stretching, tapering and twisting[41, 8]; and
- computing the volume and surface area of a swept volume[7, 35].

This effort has also led to a collection of mathematical approaches, including sweep differential equation[8], Jacobian rank defiency method[1], sweep envelop theory[19, 22], and manifold stratification[2].

A noticeable amount of the previous work on rendering swept volumes focused on ray tracing specific classes of sweep representations. These classes include translational and rotational sweeps[26, 44], sweeps with a sphere template[45], and generalised cylinders[13, 14, 15], which are constructed by sweeping a 2D contour along a 3D a parametric curve. Generalised cylinders are also considered as a family of parametric models, especially when their 2D contours are also defined using parametric curves. Unfortunately, it is sometimes difficult to obtain the inverse of a sweeping rule that can be solved analytically. In such cases, numerical techniques, which are often computationally intensive, are employed to determine the intersection between a ray and a generalised cylinder.

Instead of ray tracing swept volumes directly, it is often more convenient to convert them to alternative representations, such as polygonal meshes, prior to rendering[12, 40, 25, 6]. One relatively common approach is to voxelise a swept object into a 3D regular grid as an intermediate representation[12, 40], from which an iso-surface is then computed[31]. Recently, Sealy and Wyvill described an adaptive algorithm for subdividing a trajectory during the process of voxelising a general swept object[41]. Aguado and Zaluska described a method for modelling a generalised cylinder by converting a 2D contour to a Fourier descriptor and sweeping the descriptor using Fourier morphing[6]. Wu et al. also considered a voxelisation approach for sweeping an image along a NURBS curve whilst assuming no self-intersection[47]. These provided the authors with the initial motivation for developing volume sweeping techniques using *volume data types*[33].

Volume data types are representations that describe objects in a true three-dimensional manner. The underlying mathematical notion of such a data type is a set of scalar fields that define some geometric and physical attributes of every point in Euclidean space. Early volume data types include binary spatial representations such as spatial occupancy enumeration and constructive solid geometry[36]. Real domain volume data types have also been deployed effectively in schemes such as implicit surfaces[11], solid and hyper-textures[34]. Videos can also be considered as a volume data type[28].

Over the past two decades, we have also witnessed the rapid development of volume visualisation techniques[31, 30, 38, 43], driven mainly by applications such as medical imaging and scientific computation. These developments suggest that volume-based techniques may have the potential to match, and in some aspects supersede, surface-based techniques in computer graphics[16, 27]. This work is intended to explore further the use of volume-based modelling and rendering techniques.

3. Concepts of Image-Swept Volumes

Consider an $n_x \times n_y$ image, which is mapped onto a *normalised planar sweeping template*:

$$\mathbf{m}(x, y), \qquad 0 \leq x, y \leq 1$$

using an interpolation function. Although a *sweeping trajectory* may be defined in a variety of ways, we conveniently use a parametric function to represent a 3D trajectory:

$$\mathbf{a}(u) = (a_x(u), a_y(u), a_z(u)), \qquad u_{min} \leq u \leq u_{max}$$

When the parameter u proceeds from u_{min} to u_{max}, $(a_x(u), a_y(u), a_z(u))$ traces out the trajectory. Let $\mathbf{r}(x, y, u)$ be a function of affine transformation that changes the orientation and size of the template $\mathbf{m}(x, y)$ along the trajectory $\mathbf{a}(u)$. A *swept volume* can thus be defined as an infinite set $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$, which contains every instance of mapping from a point (x, y) in the template $\mathbf{m}$ to a point p in the volume. In each instance, a point p is associated with a value v. Each instance of mapping can be expressed as a tuple:

$$\gamma_i = < p_i, \ v_i >$$

A function can thus be used to specify Γ as:

$$\gamma(x, y, u) = < \mathbf{r}(x, y, u) + \mathbf{a}(u), \ \mathbf{m}(x, y) >$$
$$= < \rho(x, y, u), \ \mathbf{m}(x, y) >$$

where $\rho(x, y, u)$ defines a forward mapping function from the sweeping space to $\mathbb{E}^3$. Note that $\rho(x, y, u)$ does not guarantee a one-to-one mapping between an instance $\gamma \in \Gamma$ and a point $p \in \mathbb{E}^3$. A many-to-one mapping at p implies a self-intersection, which would impair many existing solutions to construction and display of swept volumes. We will discuss our method for dealing with self-intersections in Sections 4 and 5. In Section 5, we will also extend the above concept to consider a sweep involving a set of different templates, which are typically represented by an $n_x \times n_y \times n_z$ volume dataset.

By utilising an appropriate real-domain operator η for combining values mapped to the same p, we construct a *scalar field* in Euclidean space $\mathbb{E}^3$ as:

$$F(p) = \begin{cases} v & \text{a unique } \gamma \text{ in } \Gamma \text{ with } \rho(x,y,u) = p \\ \eta(v_1, \ldots) & \text{multiple } \gamma\text{'s in } \Gamma \text{ with } \rho(x,y,u) = p \\ c & \text{no } \gamma \text{ in } \Gamma \text{ such that } \rho(x,y,u) = p \end{cases}$$

where c is a predefined constant.

A scalar field $F(p)$ offers a true 3D representation that defines data for every point p in $\mathbb{E}^3$. One can effectively use one or a group of scalar fields to specify the exterior of an object (e.g., blobby model[11]), interior of an object (e.g., colour volume[48] and 3D texture[34]), and an amorphous phenomenon (e.g., clouds[23]).

Also note that in the context of sweep representations the term "volume" normally refers to a continuous volume bounded by a surface. However, in volume visualisation, "volume" is often used exchangeably with "volume dataset" suggesting a collection of discrete voxels. In the following discussions, we use "volume" to imply only a bounded domain in $\mathbb{E}^3$, and "volume dataset" for a finite collection of discrete voxels.

A *spatial object* is a tuple, $\mathbf{o} = (O, A_1, \ldots, A_k)$, of scalar fields defined in $\mathbb{E}^3$, including an *opacity* field $O : \mathbb{E}^3 \to [0,1]$ specifying the "visibility" of every point p in $\mathbb{E}^3$, and possibly other *attribute* fields $A_1, \ldots, A_k : \mathbb{E}^3 \to \mathbb{R}, k \geq 0$.

The opacity field, which specifies the "visible geometry" of a spatial object, is the most elementary field in volume modelling. Any point which is not transparent is potentially visible to a rendering algorithm. Attribute fields are used to define other properties of a spatial object, such as colours, reflection coefficients, refraction indices, and so on. More comprehensive discussions on modelling spatial objects using multiple scalar fields can be found in two recent articles[17, 18]. Without losing generality, we illustrate the use of only four attribute fields, including three colour channels $R, G, B : \mathbb{E}^3 \to [0,1]$ and an optional geometry field $G_{eo} : \mathbb{E}^3 \to [0,1]$ over which a normal estimation algorithm, such as central difference, may compute[18].

Figure 1 shows a spatial object whose O, R, G, B, G_{eo} fields are image-swept volumes. To construct this object, we simply converted a photograph of a milk-bottle into a set of image templates using a few basic image processing filters (e.g., edge-detection, blurring, colour-filling), and then applied a rotational sweep to these templates.

4. Modelling Image-Swept Volumes

In this section, we first outline a modelling scheme for specifying a sweep. We then consider the use of an adaptive subdivision approach to the construction of a volume dataset as an input to a typical volume rendering integral[33]. We also discuss methods for dealing with self-intersections in a sweep.

4.1. Specifying Sweeps

The specification of a sweep contains primarily a sweeping trajectory $\mathbf{a}(u)$, and an affine transformation function $\mathbf{r}(x, y, u)$ which defines a local coordinate transformation at each point along $\mathbf{a}(u)$. To facilitate a more intuitive specification of image-swept volumes, we introduce a set of user-definable functions, which include:

- *sweeping trajectory*, $\mathbf{a}(u)$ — This is a series of trajectory segments which may or may not be joined together. Each segment can be a line segment (for a simple translational sweep), a circular segment (for a rotational sweep), a parametric curve segment, or an arbitrary function returning a point $p \in \mathbb{E}^3$ for every input u.
- *sweeping direction*, $\mathbf{r_d}(u)$ — This defines the normal vector of the template at $\mathbf{a}(u)$. By default, this is the tangent to the trajectory, that is, $d\mathbf{a}(u)/du$. In addition, we allow a more complex specification of sweeping directions using a secondary trajectory $\mathbf{a_d}(u)$, with which we have $\mathbf{r_d}(u) = \mathbf{a_d}(u) - \mathbf{a}(u)$. An arbitrary function returning a vector is also allowed.
- *sweeping up-vector*, $\mathbf{r_u}(u)$ — This defines a rotation of the template about $\mathbf{r_d}(u)$ at every point along $\mathbf{a}(u)$, subject to $\mathbf{r_u}(u)$ and $\mathbf{r_d}(u)$ being discollinear. $\mathbf{r_u}(u)$ can also be conveniently specified using an additional trajectory $\mathbf{a_u}(u)$, yielding $\mathbf{r_d}(u) = \mathbf{a_u}(u) - \mathbf{a}(u)$; or using an arbitrary function returning a vector.
- *sweeping scaling factors*, $\mathbf{r_s}(u)$ — This defines the x- and y-scaling factors in the template space. By default, $\mathbf{r_s}(u) = [1,1]$. Interesting effects can be generated easily using a more complex function.

As illustrated in Figure 2, $\mathbf{r_d}(u)$ and $\mathbf{r_u}(u)$ give a specification of a local coordinate system, $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$. From $\mathbf{r_d}(u)$, $\mathbf{r_u}(u)$ and $\mathbf{r_s}(u)$, we can easily obtain a composite transformation $\mathbf{r}(u)$, which, together with $\mathbf{a}(u)$, defines a sweep.
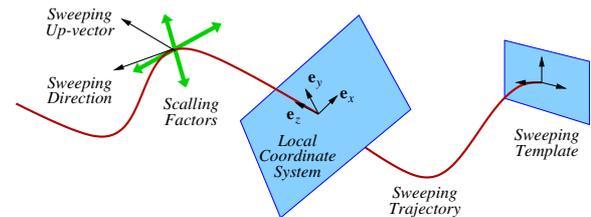

Figure 2: *The specification of a sweep.*

Figure 3 shows four example sweeps in which a character '@' is swept using different sweeping specifications. The outer ring of the letter is translucent and coloured in cyan, while the inner '*a*' is opaque, and coloured in red. The images were synthesised using volume ray casting[30].
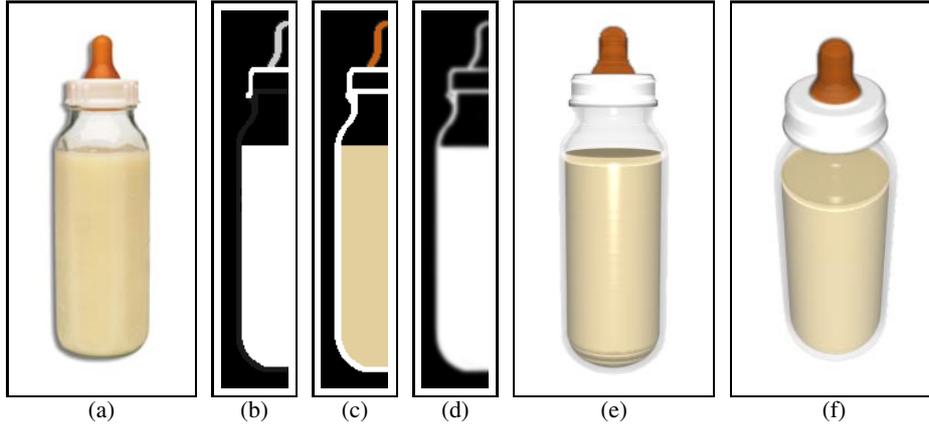
Figure 1: *A milk-bottle object constructed from five image templates. They are: (a) a photograph of a milk-bottle, from which image templates are defined; (b) an opacity template; (c) an image representing three colour templates; (d) a geometry template; (e) a milk-bottle object constructed using a rotational sweep; and (f) a different view of the object.*

4.2. Voxelising Image-Swept Volumes

It is desirable to render a sweep in its original specification if the inverse mapping of $\rho(x, y, u)$ can be obtained algebraically. We shall examine the direct rendering approach in detail in Section 5.

However, it is often difficult to obtain an algebraic solution of the inverse mapping, or an accurate numerical solution at a reasonable computational cost. In such a case, hence, it would be more cost-effective to approximate a sweep specification using a different graphical representation. As discussed in Section 3, a swept volume is essentially a bounded scalar field, which can be approximated by a volume dataset. The most commonly-used form of volume datasets is a 3D regular grid, where a voxel is defined at each grid point and is associated with a value or a set of values. Such a dataset can be rendered using a range of volume rendering techniques[31, 30, 38, 43].

Volume datasets were used as an output in the previous work on swept volumes by Sealy and Wyvill[41] and by Wu et al.[47] Sealy and Wyvill also outlined a subdivision algorithm for voxelising sweeps following a parametric curve, which has formed the basis of our voxelisation approach. Sealy and Wyvill considered the contour-based templates, while the voxelisation method by Wu et al. is limited to sweeps without self-intersections. In this and the following sections, we will combine these two methods into an algorithm for voxelising a range of image-swept volumes.

Consider a *composite trajectory* $\mathbf{a}(u) = \{\mathbf{a}_i(u_i); u_{i,min} \leq u_i \leq u_{i,max}; i = 1, 2, \ldots\}$. For each trajectory segment $\mathbf{a}_i(u_i)$, we recursively subdivide the segment in its parameter space $[u_{i,min}, u_{i,max}]$ until the distance between two neighbouring templates has reached a pre-defined threshold τ. Note that τ is defined in $\mathbb{E}^3$ (or in relation to the size of the volume dataset) rather than in the parameter space. Hence the subdivision process is adaptive to the resolution set in $\mathbb{E}^3$ though it is controlled within the parameter space.

At each sub-segment defined by $[u_{i,a}, u_{i,b}], u_{i,min} \leq u_{i,a} < u_{i,b} \leq u_{i,max}$, we obtain two forward-mapped templates $\mathbf{m}_{i,a}$ and $\mathbf{m}_{i,b}$ at $u_{i,a}$ and $u_{i,b}$ respectively. For each voxel bounded by the two templates, we project the voxel onto each of the templates along the corresponding sweeping direction. Each template is then sampled at the projection point, and the two sampled values are linearly interpolated to yield the voxel value. For a spatial object with multiple scalar fields, multiple templates are sampled simultaneously.

However, it is necessary to take care of the situation when the two mapped templates, $\mathbf{m}_{i,a}$ and $\mathbf{m}_{i,b}$, intersect with each other. Such a condition may split the region between $\mathbf{m}_{i,a}$ and $\mathbf{m}_{i,b}$ into two disjoint sub-regions. In our implementation, a 3D boundary-filling algorithm is adapted for scan-converting regions between mapped templates, and potential splitting cases are dealt with by using multiple seeds.

Objects in Figure 3 are rendered from their voxelised representations. The quality of the approximation depends mainly on the τ value and the resolution of the volume dataset to be generated. The speed depends mainly on the complexity of the trajectory concerned and the size of the volume dataset, and marginally on the τ value. For each of the objects in Figure 3, it took less than two minutes on a 400 MHz PC to generate a set of five $150 \times 150 \times 300$ volume datasets representing the O, R, G, B, G_{eo} scalar fields.

4.3. Dealing with Self-Intersections

Self-intersections in a sweep may occur in many ways. It could simply be a case that a trajectory is self-intersecting as one of the objects in Figure 3. It can also be caused by have sweeping templates colliding with each other while
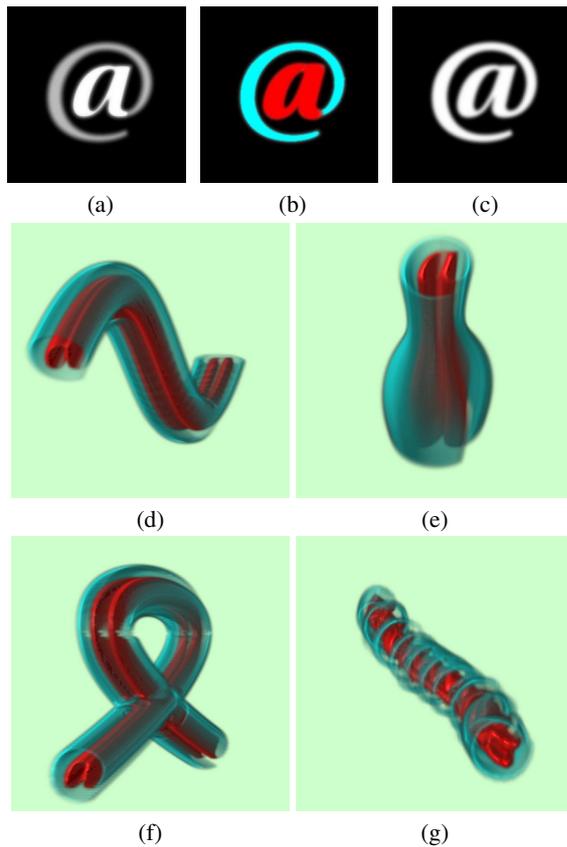
Figure 3: *Examples of image-swept objects, which were voxelised prior to the rendering: (a) the opacity template of a letter '@'; (b) an image representing the three colour templates of the @; (c) the geometry template of the @; (d) a sweep following a Bézier curve; (e) a sweep with varying template sizes; (f) a sweep containing self-intersections; and (g) a sweep with varying template orientations.*

travelling along different trajectory segments, as shown in Figure 4. If we were sweeping a solid volume with a homogeneous interior, we would naturally apply the Boolean "union" operator to the intersected parts of the sweep. However, for image-swept volumes, we must employ combinational operators in the real domain, and consider different operators for different attribute fields.

Chen and Tucker recently proposed an algebraic framework for Constructive Volume Geometry (CVG)[17], which is well suited for dealing with self-intersections in image-swept volumes. CVG is a major generalisation of CSG (Constructive Solid Geometry). Unlike CSG, CVG does not limit itself to geometrical operations only. It utilises the notion of spatial objects as defined in Section 3 and can also be employed to manipulate all attribute fields associated with

objects. Its combinational operations, mostly defined in the real domain, can be used to model complex internal structures of objects and amorphous phenomena in a constructive manner.

In CVG, operations on spatial objects are decomposed into those on scalar fields, which are further decomposed into those on scalars. Figure 4 shows an object that was constructed by sweeping the "@" template (as in Figure 3) along four trajectory segments. In two separate sweeps, a real-domain "union" operation and a blending operation were applied to the intersected parts respectively. In this case, the "blending" operation gives the swept object a more natural look. CVG also offers other real-domain operators such as "intersection" and "difference", which may be used to specify the effect of an intersection in an unconventional way.

In comparison with the traditional solutions to the intersection problem, we have shifted the main focus from the identification of a new boundary representation to the intermixing of the intersected parts of a swept object. A high-level representation in the form of a CVG term can be forwarded directly to a CVG-enabled renderer (such as *vlib*[46]), or be voxelised into a single volume object that can be handled by any volume renderers. Should the identification of an explicit boundary be required, an isosurfacing algorithm can be employed[31]. Nevertheless, such a step is neither necessary nor desirable in a volume graphics pipeline. Further details about modelling and rendering CVG objects can be found in the literature[17].

5. Rendering Image-Swept Volumes

In the section, we first briefly outline a general approach to the rendering of spatial objects. We then consider the direct rendering of image-swept volumes with the aid of a simple sweeping specification as an example. This is followed by a discussion on dealing with a more complex sweeping trajectory, namely Bézier curves. Finally, we illustrate the use of a collection of image templates, such as a video, to generate an interesting sweep.

5.1. Discrete Ray Tracing

As discussed in Section 3, image-swept volumes are essentially a specific type of spatial object. They can be integrated with other spatial objects into a graphics scene, which can then be rendered directly using a multi-object renderer or voxelised into a single volume dataset. Discrete ray tracing[30, 42] is the most fundamental method for rendering volume datasets directly. The method was also generalised in recent years to handle multiple volume objects in a complex scene[29, 17], and in particular *vlib*[46], a general purpose volume graphics API, provides an open source multi-object rendering engine.
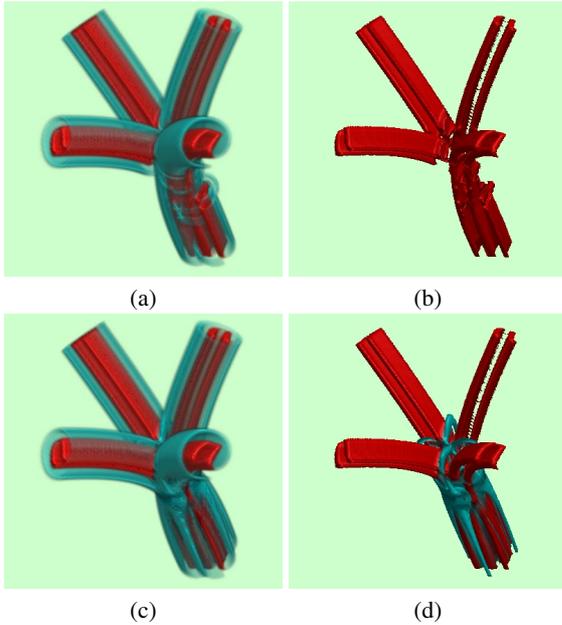
As part of our implementation, we have incorporated a

Figure 4: *Objects swept along a trajectory consisting of four segments which are disjoint at the ends but intersected with each other on their sweeping paths. (a)-(b) a real domain "union" operation is used to resolve the intersections, with (a) shows the swept object, and (b) shows an iso-surface at opacity = 0.9. (c)-(d) a real domain blending operation is used to resolve the intersections, with (a) shows the swept object, and (b) an iso-surface at opacity = 0.9.*

class of image-swept volumes into *vlib*. In *vlib*, spatial objects (including swept objects and composite objects) are each associated with a bounding box. The direct rendering in *vlib* is based on a ray tracing approach. For each ray, intersections between the ray and object bounding boxes are computed, and intersected objects are processed in the order of proximity. Within each bounding box, discrete samples are taken along the ray and appropriate rendering operations such as normal estimation, shading and opacity/colour accumulation, are performed. Unlike traditional surface-based ray tracing, complex tests for ray-sweep intersections are not necessary. The bounding box of an object also facilitates a transformation from the *world coordinate system* to the corresponding *local object coordinate system*. In addition to image-swept volumes, *vlib* also includes many other modelling and rendering features such as constructive volume geometry, solid and hyper-textures, shadows, reflection and refraction. Figure 5 shows a scene modelled and rendered using *vlib*, and the scene includes two objects specified as image-swept volumes (i.e, the glass and the bottle). A texture is mapped onto the bottle to demonstrate the feasibility of integrating image-swept volumes with other graphics techniques.

5.2. Direct Evaluation of Simple Sweeps

To render a swept volume directly in its original specification, we need to evaluate its corresponding scalar fields, $F(p)$, as defined in Section 3. In order words, given a forward sweeping function $\rho(x, y, u)$, we have to obtain its inverse mapping either algebraically or numerically.

Figure 5: *Image-swept volumes in a multi-object scene.*

Consider an arbitrary point $p \in \mathbb{E}^3$. A rendering algorithm must identify all instances in Γ such that $p = \rho(x, y, u)$. For each of these instances, we have its inverse mapping as:

$$[x, y, u] = \rho^{-1}(p)$$

From $[x, y, u]$, v can be determined trivially from the template. Should there be more than one instance, an appropriate scalar operation η is applied to $v_1, v_2, \dots$ to obtain a composite scalar value. It is not always necessary to obtain u, unless it must be computed in order to derive (x, y) or it is used to sample a volume template as discussed in Section 5.4.

A sweep is considered to be *simple* if a one-to-one or many-to-one inverse mapping $\rho^{-1}(p)$ can be obtained easily. For example, the swept volume in Figure 1 was defined with a rotational sweeping specification:

$$\mathbf{a}(u) = [r \cdot \cos(u), 0, r \cdot \sin(u)], \qquad 0 \le u \le 2\pi$$

where we assume that the template rotates about the y-axis.

In our implementation, as mentioned in 5.1, the evaluation of a sweeping specification takes place in a local object coordinate system. In other words, without losing generality, we assume that each sampling p within the bounding box of a swept object is transformed to the local sweeping space prior to the inverse mapping.

Thus, the inverse mapping of this rotational sweeping is:

$$x = \sqrt{p_x^2 + p_z^2}, \qquad y = p_y,$$

$$u = \begin{cases} \arctan(p_z/p_x) & p_x > 0 \\ \pi + \arctan(p_z/p_x) & p_x < 0 \\ 0 & p_x = p_z = 0 \\ \text{sign}(p_z)\pi/2 & p_x = 0, p_z \neq 0 \end{cases}$$

where it is necessary to evaluate the validity of the obtained (x, y), that is, testing (x, y) for inclusion in the template.

Note that, for each p on the y-axis, there is an infinite number of instances. This is a singularity condition which is normally very difficult to deal with. Fortunately, in this particular sweep, all instances for such a p are exactly the same, and all other points are associated with a one-to-one mapping.

5.3. Direct Evaluation of a Sweep over a Bézier Trajectory

For a more general sweep, we can define the sweeping trajectory using an n-degree Bézier curve:

$$\mathbf{a}(u) = \sum_{i=0}^{n} C_i B_n^i(u), \qquad 0 \leq u \leq 1$$

where C_i represents the $n + 1$ Bézier control points and B_n^i is the Bernstein-basis function:

$$B_n^i(u) = \frac{n! \, u^i (1-u)^{n-i}}{(n-i)! \, i!}$$

Let p be an arbitrary point and $\mathbf{v}$ be the vector formed by $\mathbf{a}(u) - p$. We know that point p lies co-planar with some sweeping template $\mathbf{m}(x, y)$ at u only if $\mathbf{b}$ forms a right angle with the tangent vector at $\mathbf{a}(u)$, that is,

$$\mathbf{v} \bullet \frac{d\mathbf{a}(u)}{du} = 0$$

We can solve this equation to find the candidate u values. Note that for curves with $n \geq 3$, it is impossible to derive a general solution analytically, given that $\mathbf{v} \bullet (d\mathbf{a}(u)/du)$ expands to a polynomial of at least degree 5. In this case, it is necessary to employ a numerical root finder such as that described in[39].

The next step is to determine an index to the template corresponding to each solution of u. It is probable that many sampling points taken along a ray, even though one or more u values may be computable, will not actually intersect with the template as it is swept along the Bézier trajectory. For such sampling points, we have $F(p) = c$ as defined in Section 3.

It is also likely that a sampling point will intersect with the template at only some, but not all, of the computed u values. A substantial amount of computation will be saved

if a u value can be trivially rejected on the basis that a sampling point is far enough away from $\mathbf{a}(u)$ not to lie inside the template as it is swept along this part of the trajectory. If the template is bounded by a circle with radius r, as illustrated in Figure 6, then the sampling point p may be rejected if it is further than r distance away from $\mathbf{a}(u)$. Hence, a computed u value can be safely rejected if:

$$|\mathbf{v}| > 0.5\sqrt{(\mathbf{r_{sx}}(u))^2 + (\mathbf{r_{sy}}(u))^2}$$

If the scaling factors, $\mathbf{r_{sx}}(u)$ and $\mathbf{r_{sy}}(u)$, are constant over the trajectory then the right hand side of this inequality can be precomputed.
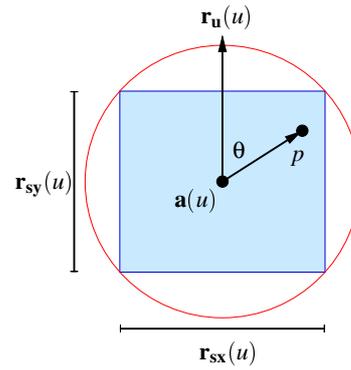

Figure 6: *A cross-section of the Bézier trajectory.*

If all computed u values are rejected, we know that the sampling point does not intersect with the image-swept volume and therefore the evaluation at this point can terminate. For the remaining u values, an *up* vector and a *right* vector must be determined for the template so that a local coordinate system can be established. The sampling point can then be mapped into this local coordinate system and a template index consequently determined. The *right* vector is the binormal vector to the curve. A binormal vector, $\mathbf{b}$, is perpendicular to the direction vector and is computed with:

$$\mathbf{b} = \mathbf{r_d}(u) \times \frac{d\mathbf{r_d}(u)}{du}.$$

The up vector may then be computed simply from the right and direction vectors to form a triad:

$$\mathbf{r_u}(u) = \mathbf{r_d}(u) \times \mathbf{b}.$$

$\rho^{-1}(p)$, the mapping from object space to template space, and therefore an index to the normalised sweeping template, may then be computed in terms of $\mathbf{v}$, $\mathbf{b}$, and $\mathbf{r_u}(u)$, with:

$$\rho^{-1}(p) = \left(0.5 + \frac{|\mathbf{v}|}{\mathbf{r_{sx}}(u)}\sin\theta, \; 0.5 - \frac{|\mathbf{v}|}{\mathbf{r_{sy}}(u)}\cos\theta, \; u\right)$$

where θ is the angle between $\mathbf{v}$ and $\mathbf{r_u}(u)$.

Figure 7 shows an image-swept volume created by sweeping an image template along a cubic Bézier curve, and rendered by direct evaluation.
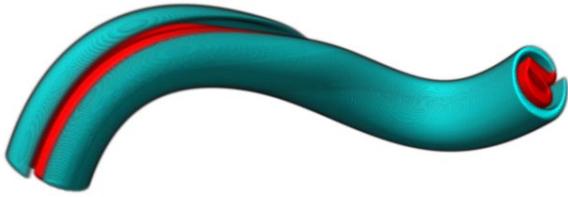
Figure 7: *An image-swept volume rendered by direct evaluation with a numerical root finder. The sweeping trajectory is defined by a cubic Bézier curve.*

5.4. Multiple Image Templates

So far, we have described how a field in a volume object may be defined by sweeping a single image template along an arbitrary trajectory, $\mathbf{a}(u)$. We now show, by way of example, how multiple image templates may be employed to model a single field in order to produce non-uniform volume objects.

Given a number of different images, we can organise them into an $n_x \times n_y \times n_z$ volume dataset, which can be mapped onto a *normalised volume sweeping template*:

$$\mathbf{w}(x,y,z), \qquad 0 \le x,y,z \le 1$$

using an interpolation function. The value z is typically defined by an *image path-association function* $\mathbf{f}(u)$.

For example, the "ring of fire" object in Figure 8 was constructed using such a volume template. The template was built from a video of a small flame, which consists of 256 frames. The sweeping trajectory of the templates is defined by a simple circular path:

$$\mathbf{a}(u) = [3 \cdot \cos(u), 0, 3 \cdot \sin(u)], \qquad 0 \le u \le 2\pi$$

and the image-path association function used is:

$$\mathbf{f}(u) = \frac{u \cdot 255}{2\pi}$$

The similarity between successive frames in the original video have resulted in a relatively smooth and realistic fire object. The amorphous nature of the original video has been effectively retained in the swept volume. Figure 9 shows another two examples of sweeps using the same video. Whilst these examples have demonstrated the use of multiple image templates in modelling swept volumes, they have also echoed the suggestion by Klein et al. for deploying video data in general computer graphics[28].

6. Conclusions

We have presented a novel approach for constructing a swept volume using one or more image templates. Also, we have described both the voxelisation of an image-swept volume in modelling space, and the direct evaluation of a sweeping specification in rendering space. To summarise, our new contributions to the field are as follows:

Figure 8: *A large ring of fire is created from a small flame by using a rotational sweep and a volume template.*

Figure 9: *Two different sweeps that use the same volume template as in Figure 8, but involve additional geometrical transformations.*

- We have focused our work on a true 3D representation of a swept volume featuring heterogeneous interiors and amorphous effects, whilst most previous studies have focused on solid objects with well-defined contour templates;
- We have studied a range of image-based sweeping specifications, including multi-channel image templates, video templates, generalised sweeps, and self-intersecting trajectories;
- We have utilised an adaptive subdivision method for generating voxelised approximations, and in particular, we have employed CVG operators to deal with multiple and self-intersecting trajectories;
- Within a ray-tracing framework, we have developed direct evaluation methods for a class of image-swept volumes, including those swept over an n-degree Bézier curve;
- We have also extended the concept to include multiple

image templates, which facilitates the use of 3D volume datasets such as medical data and those derived from video.

Our study has shown that image-based sweeping can be used effectively to create a range of interesting objects, many of which are difficult to model using current techniques. For a certain class of image-swept volumes, direct evaluation during rendering can yield higher quality results than is otherwise possible. However, voxelisation remains to be a useful tool for handling sweeping specifications for which an inverse mapping is difficult to obtain.

In conclusion, we believe that the concept of image-swept volumes will offer itself as a useful and practical technique for modelling complex volumetric objects, in particular, amorphous phenomena and objects with heterogeneous interiors. This technique has the potential to serve a range of applications where traditional surface-based volume sweeping has a limited role.

References

1. K. Abdel-Malek and H. J. Yeh, "Geometric representation of the swept volume using Jacobian rank deficiency conditions", *Computer-Aided Design*, **29**(6), pp. 457–468, (1997).

2. K. Abdel-Malek, H. J. Yeh and S. Othman, "Swept volumes: void and boundary identification", *Computer-Aided Design*, **30**(13), pp. 1009-1018, (1998).

3. K. Abdel-Malek, D. Blackmore and K. Joy, "Swept Volumes: Foundations, Perspectives, and Applications", submitted to *International Journal of Shape Modeling*, (2000).

4. K. Abdel-Malek and S. Othman "Multiple sweeping using the Denavit-Hartenberg representation method", *Computer-Aided Design*, **31**, pp. 567–583, (1999).

5. K. Abdel-Malek, J. Yang and D. Blackmore, "Closed-form swept volume of implicit surfaces", *Proc. the 26th ASME Design Automation Conference*, Baltimore, MD, (2000).

6. A. S. Aguada and E. Montiel, "Modeling generalized cylinders via Fourier morphing", *ACM Transactions on Graphics*, **18**(4), pp. 293–315, (1999).

7. M. J. al-Daccak and J. Angeles, "Reduced calculation of volumetric properties of sweep-generated solids", *ASME DE Computer-Aided and Computation Design – Advances in Design Automation*, **19**(part 1), pp. 157–162, (1989).

8. D. Blackmore, M. C. Leu and L. P. Wang, "Analysis and modeling of deformed swept volumes", *Computer-Aided Design*, **26**(4), pp. 315–326, (1994).

9. D. Blackmore, M. C. Leu, L. P. Wang and H. Jiang, "Swept Volumes: a retrospective and prospective view", *Neural, Parallel and Scientific Computations*, **5**, pp. 81–102, (1997).

10. D. Blackmore, R. Samulyak and M. C. Leu, "Trimming swept volumes", *Computer-Aided Design*, **31**(3), pp. 215–223, (1999).

11. J. F. Blinn, "A generalization of algebraic surface drawing", *ACM Transactions on Graphics*, **1**(3), pp. 235–256, (1982).

12. J. Bloomenthal, "Polygonization of implicit surfaces", *Computer Aided Geometry Design*, **5**(4), pp. 341–355, (1988).

13. W. F. Bronsvoort and F. Klok, "Ray tracing generalized cylinders", *ACM Transactions on Graphics*, **4**(4), pp. 291–303, (1985).

14. W. F. Bronsvoort, P. R. van Nieuwenhuizen and F. H. Post, "Displayed of profiled sweep objects", *Visual Computing*, **5**, pp. 147–157, (1989).

15. W. F. Bronsvoort, "A surface-scanning algorithm for displaying generalized cylinders", *Visual Computing*, **8**(3), pp. 162-170, (1992).

16. M. Chen, A. E. Kaufman and R. Yagel (eds), *Volume Graphics*, Springer, London, (2000).

17. M. Chen and J. V. Tucker, "Constructive Volume Geometry", *Computer Graphics Forum*, **19**(4), pp.281–293, (2000).

18. M. Chen, A. S. Winter, D. Rodgman and S. M. F. Treavett, "Enriching volume modelling with scalar fields", to appear in F. Post, G.-P. Bonneau and G. Nielson (eds) *Volume Visualization*, Kluwer Academic Press (2002).

19. B. Dahlberg and B. Johansson, "Envelope curves and surfaces", *The Mathematics of Surfaces II*, University Press, pp. 419–426, (1987).

20. B. Fleming, *3D Modeling and Surfacing*, Morgan Kaufmann, (1999)

21. J. D. Foley, A. van Dam, S K. Feiner and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, (1990).

22. M. A. Ganter and J. Uiker, "Dynamic collision detection using swept solids", *ASME Journal of Mechanism, Transmission, and Automation in Design*, **108**, pp. 549–555, (1986).

23. G. Y. Gardner, "Simulation of natural scenes using textured quadric surfaces", *ACM/SIGGRAPH Computer Graphics*, **18**(3), pp. 11–20, (1984).

24. R. Jakubowski, "Multiple sweeping in product modeling and analysis", *Proc. the 1993 ASME Winter Annual Meeting*, New Orleans, LA, **6**, pp. 125–133, (1993).

25. K. I. Joy and M. A. Duchaineau, "Boundary determination for trivariate solids", *Proc. the 1999 Pacific Graphics Conference*, Seoul, Korea, (1999).

26. J. T. Kajiya, "New techniques for ray tracing procedurally defined objects", *ACM/SIGGRAPH Computer Graphics*, **17**(3), pp. 91–102, (1983).

27. A. Kaufman, D. Cohen and R. Yagel, "Volume graphics", *IEEE Computer*, **26**(7), pp. 51–64, (1993).

28. A. Klein, P.-P. Sloan, A. Colburn, A. Finkelstein and M. F. Cohen, *Video Cubism*, Microsoft Research TR-2001-45, (2001).

29. A. Leu and M. Chen, "Modelling and rendering graphics scenes composed of multiple volumetric datasets", *Computer Graphics Forum*, **18**(2), pp. 159–171, (1999).

30. M. Levoy, 'Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, **8**(5), pp. 29–37, (1988).

31. W. Lorensen and H. Cline, "Marching cubes: a high resolution 3D surface construction algorithm", *ACM/SIGGRAPH Computer Graphics*, **21**(4), pp. 163–169, (1987).

32. C. Madrigal and K. I. Joy, "Generating the envelope of a swept trivariate solid", *Proc. the IASTED International Conference on Computer Graphics and Imaging*, Palm Springs, CA, (1999).

33. G. Nielson, "Volume modelling", in M. Chen, A. Kaufman and R. Yagel (eds), *Volume Graphics*, pp. 29–48, Springer, London, (2000).

34. K. Perlin, "An image synthesizer", *ACM/SIGGRAPH Computer Graphics*, **19**(3), pp. 287–296, (1985).

35. J. Peters and A. Nasri, "Computing volumes of solids enclosed by recursive subdivision", *Computer Graphics Forum*, **16**(3), pp.89–94, 1997.

36. A. A. G. Requicha, "Representations for rigid solids: theory, methods and systems", *Computing Surveys*, **12**(4), pp. 437–464, (1980).

37. A. A. G. Requicha and H. B. Voelcker, "Solid modeling: a historical summary and contemporary assessment", *IEEE Computer Graphics and Applications*, **2**(2), pp. 9–24, (1982).

38. P. Sabella, "A rendering algorithm for visualizing 3D scalar fields", *ACM/SIGGRAPH Computer Graphics*, **22**(4), pp. 51–58, (1988).

39. P. J. Schneider, "A Bezier curve-based root-finder" A. S. Glassner (eds), Graphics Gems, pp. 409–415, (1990).

40. W. J. Schroeder, W. E. Lorensen and S. Linthicum, "Implicit modeling of swept surfaces and volumes", Proc. the 1994 IEEE Visualization Conference, Washington, DC, pp. 40–45, (1994).

41. G. Sealy and G. Wyvill, "Representing and rendering sweep objects using volume models", *Proc. the 1997 International Conference on Computer Graphics*, Hasselt, Belgium, pp. 22–27, (1997).

42. R. Yagel, D. Cohen and A. Kaufman, "Discrete Ray Tracing", *IEEE Computer Graphics and Applications*, **12**, pp. 19–28, (1992).

43. L. Westover, 'Footprint evaluation for volume rendering", *ACM/SIGGRAPH Computer Graphics*, **24**(4), pp. 59–64, (1988).

44. J. J. van Wijk, "Ray tracing objects defined by sweeping planar cubic splines", *ACM Transactions on Graphics*, **3**(3), pp. 223-237, (1984).

45. J. J. van Wijk, "Ray tracing objects defined by sweeping a sphere", *Proc. Eurographics '84 Conference*, Copenhagen, pp. 73–82, North-Holland, Amsterdam, (1984).

46. A. S. Winter and M. Chen, "vlib: A Volume Graphics API", in K. Mueller and A. Kaufman (eds.), *Volume Graphics 2001*, pp. 133–147, Springer-Wien New York (2001), *see also http://vg.swan.ac.uk/vlib/*.

47. Z. Wu, H. S. Seah and F. Lin, "NURBS volume for modelling complex objects", in M. Chen, A. Kaufman and R. Yagel (eds), *Volume Graphics*, pp. 97–117, Springer, London, (2000).

48. G. Wyvill and P. Sharp, "Volume and surface properties in CSG", in *Proceedings of Computer Graphics International '88: New Trends in Computer Graphics*, pp. 257–266, Springer-Verlag, (1988).