Function Based Flow Modeling and Animation

Ergun Akleman, Zeki Melek and Jeff S. Haberl*

July 14, 2006

Abstract

This paper summarizes a function-based approach to model and animate 2D and 3D flows. We use periodic functions to create cyclical animations that represent 2D and 3D flows. These periodic functions are constructed with an extremely simple algorithm from a set of oriented lines. The speed and orientation of the flow are described directly by the orientation and the lengths of these oriented lines. The resulting cyclical animations are then obtained by sampling the constructed periodic functions.

Our approach is independent of dimension, i.e. for 2D and 3D flow the same types of periodic functions are used. Rendering images for 2D and 3D flows are slightly different. In 2D function values directly are mapped to color values. On the other hand, in 3D function values first mapped to color and opacity and then the volume is rendered by our volume renderer.

Modeled and animated flows are used to improve the visualization of operations of rolling piston and rotary vane compressors.

Keywords: Visualization, Image Synthesis, Computer Animation, Flow Modeling and Flow Animation

1 Introduction

In this paper, our goal is to create user-controlled cyclical and variable speed flow animations. We present a function based approach for modeling and animating

*Ergun Akleman is an Assistant Professor in Visualization Sciences Program, Department of Architecture, Texas A&M University. His address: Visualization Laboratory, 216 Langford Center, College Station, Texas 77843-3137. Email: ergun@viz.tamu.edu. Zeki Melek is a Ph.D. student in Department of Computer Science at Texas A&M University. Email: melekzek@viz.tamu.edu. J. Haberl is an Associate Professor in Department of Architecture, Texas A&M University. His address: Energy Systems Laboratory, Department of Architecture, Texas A&M University, College Station, Texas 77843-3581. Email: jhaberl@esl.tamu.edu.

2D and 3D flows. Based on the new approach it is extremely simple to develop 2D or 3D flow modeling and animating systems.

The interface concept to model the flow is very user-friendly. Based on this concept we have developed systems for modeling 2D and 3D flows. We have also used the modeled 2D flow animations to improve the quality of compressor visualizations as part of a research project to investigate how multimedia can improve engineering handbooks (see acknowledgments).

A similar problem, the creation of a flow animation for a given vector field, has been popular in recent Computer Graphics publications. The most common solution to the problem is to use the line integral convolution (LIC) technique proposed by Cabral and Leedom [**?**]. Forsell and Cohen have also accomplished this task with improved periodic filters which are originally suggested by Freeman, Adelson and Heeger [**?**]. They modulated the rate of the function phase shift as a function of the vector magnitude to obtain the variable-speed flow animations [**?**, **?**]. Stalling and Hege also improved this technique by eliminating aliasing effects [**?**].

Max and Becker proposed a texture advancement technique to create cyclical flow images [**?**]. Van Gelder and Wilhelms used color tables to animate 3D flow fields [**?**]. Jobard and Leber proposed the idea of motion maps to speed up the computation of steady flow animations [**?**].

Our technique originated from mathematical foundations of implicit surfaces [**?**, **?**] and it is loosely based on functional blend operations which are widely used in morphing and implicit surface modeling. Feature-Based Image Metamorphosis (FBIM) technique suggested by Beier and Neely uses a blending operation to combine morphing effects of individual line pairs [**?**]. Similarly, Crespin also used blending operations to develop Implicit Free-Form Deformations (IFFD), which provides a framework in which most of the free-form deformation techniques can be used but is based on deformation primitives defined by a local tool and a blending function [**?**]. Blending operations are widely used in Implicit surfaces to blend to implicit surfaces. Examples of blending operations are Wywill's Bloby surfaces [**?**] and Blinn's exponential functions [**?**].

In our work, we do not use a given vector field to create flow animations. Users themselves design the flow by drawing a set of oriented lines. The speed and orientation of the flow are then described directly by the orientation and the lengths of these oriented lines. Unlike the previous research, we use an implicit-based approach instead of a line integral convolution (LIC) approach. Our solution also provides a simple approach to develop an intuitive user interface to design flow animations.

2 Methodology

Our approach is based on functions that are constructed by the operations that are used in an implicit surface construction. The flow is described by

a set of oriented lines. For each oriented line segment a periodic function is created to describe a flow in the direction and position of the line. Then, these periodic functions are combined using an approximate union operation. The following subsection describes how the periodic function is created for a given line segment.

2.1 Flow Described by One Line Segment

Let an oriented line be given by two end points v_1 and v_2. Our first goal is to construct a periodic function that will give a flow in the direction of $v_2 - v_1$ with the speed $|v_2 - v_1|$. Such a function can simply be written as

$$p(v,t) = 0.5 \sin\left(t + \omega \, \frac{(v - v_1) \bullet (v_2 - v_1)}{|v_2 - v_1|^2}\right) + 0.5, \tag{1}$$

where ω is the frequency, t is the phase and $\bullet$ scalar multiplication operator. Note that the range of this function is $[0, 1]$. Therefore, it is easy to render this function using linear interpolation.

Let the color in any point of space v in any given time t be denoted by $C(v,t)$ to render the function $p(v,t)$ we use a linear interpolation to map real numbers to colors as

$$C(v,t) = C_0(1 - p(v,t)) + C_1 p(v,t) \tag{2}$$

where $C_0 = (r_0, g_0, b_0)$ and $C_1 = (r_1, g_1, b_1)$ are two user defined colors. An example of the rendering of the function $p(v,t)$ is shown in Figure **??**. This figure also demonstrates the effect of ω.

By simply changing t between 0 and 2π in equal intervals, we obtain a set of image frames. When these frames are viewed as a cyclic animation they give an illusion of a flow in the direction of $v_2 - v_1$ with the speed $|v_2 - v_1|$.

Unfortunately, the periodic function $p(v,t)$ does not indicate the position of the line (Any line with the same direction and length create similar images). Therefore, in order to visually emphasize the line that creates the motion we use a function $d(v)$ that gives minimum distance [**?**] to the given line segment. The function $d(v)$ is given by the equality

$$d(v) = \begin{cases} |v - v_1| & \text{if } (v_1 - v_2) \bullet (v - v_1) \geq 0, \\ |v - v_2| & \text{if } (v_2 - v_1) \bullet (v - v_2) \geq 0, \\ z & \text{otherwise,} \end{cases}$$

where

$$z = |(v - v_1) \times \frac{(v_2 - v_1)}{|v_2 - v_1|}|.$$

Figure **??** shows loci curves which are specified as $d(v) = c$ where c is a positive real number that indicates the distance from the line segment.

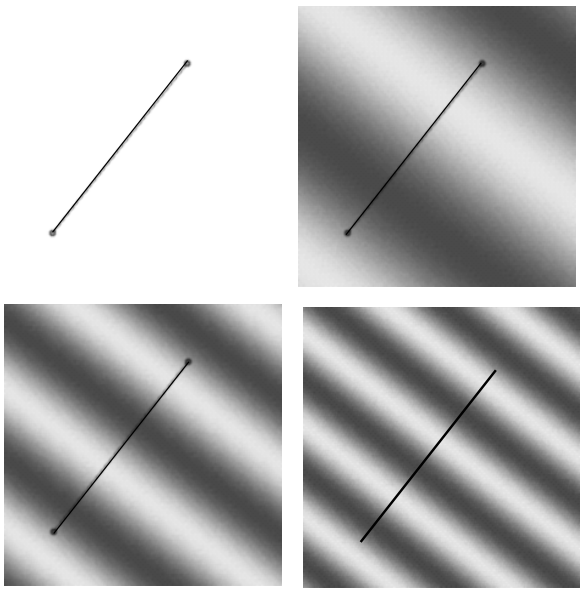

Figure 1: Periodic function for an oriented line segment. ω values are 2π, 4π and 6π.

Figure 2: Equidistant curves described by a distance function for an oriented line segment.

Based on the function $d(v)$, we introduce a parameter

$$s = \frac{d(v)}{r|v_2 - v_1|}$$

where r is a scale factor. The parameter s is 0 on the line and is 1 at $r|v_2 - v_1|$. Using this parameter, we introduce the amplitude (or opacity) function

$$a(v) = \begin{cases} (1-s)^3 + 3s(1-s)^2 & \text{if } s \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The amplitude function $a(v)$ is a C^1 function. It takes value 1 on the line segment and 0 for every point whose distance is larger than $|v_2 - v_1|/r$ to the line segment.

In these equations $p(v,t)$ gives the flow and $a(v)$ gives the shape of the line. A flow described by one line segment can threfore be described using a combination of these two functions $p(v,t)$ and $a(v)$.

2.2 Opacity and Color Functions

To render the flow we map function values to opacity and color. Let color and opacity in any point of space v in any given time t be denoted by $C(v,t)$ and $O(v,t)$ respectively.

The color $C(v,t)$ is computed using the linear interpolation we introduced before in equation **??**. For the computation of opacity $O(v,t)$ we introduce a new function which is a combination of the functions $p(v,t)$ and $a(v)$ as

$$f(v,t) = (1-u)a(v)p(v,t) + ua(v), \tag{3}$$

where u is a real number between 0 and 1. The effect of r and u are shown in Figures **??** and **??** respectively. If $u = 1$, the shape of the flow looks like a capsule as shown in Figure **??**. The thickness of the capsule can be controlled by the r parameter as shown in Figure **??**. For $u = 0.25$, we can get a shape that consist of blobs as shown in Figure **??**. These blobs can be further separated using smaller values of u.

As it can be seen in Figures **??**, **??**, **??**, and **??**, the same concept works for both 2D and 3D flow modeling. The only difference in handling 2D and 3D flows is in rendering.

2.3 Rendering Flow Described by One Line Segment

For rendering 2D flow, we simply sample color and opacity functions, $C(v,t)$ and $O(v,t)$, and obtain a set of image frames with associated color and opacity. By compositing these images with a background image, we obtain image frames for 2D flow animation. Figure **??** shows the frames of 2D flow animation with a

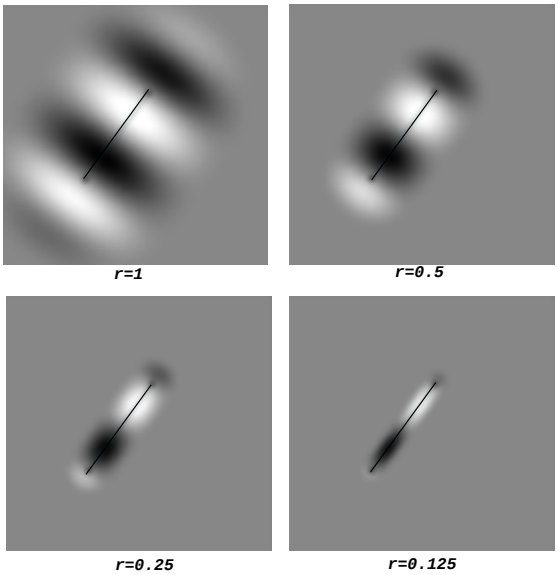

Figure 3: The effect of r ($\omega = 4\pi$ and u=0.9).

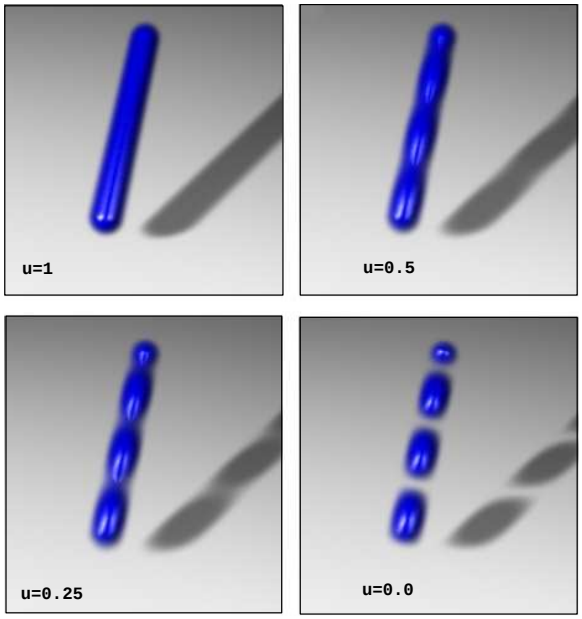

Figure 4: The effect of u ($\omega = 4\pi$, r=0.5 and $C_0 = C_1$).

gray background image. Only 4 frames are enough to create an animation. In order to avoid temporal aliasing the functions need to be sampled with a spatial and temporal jittering [**?**].

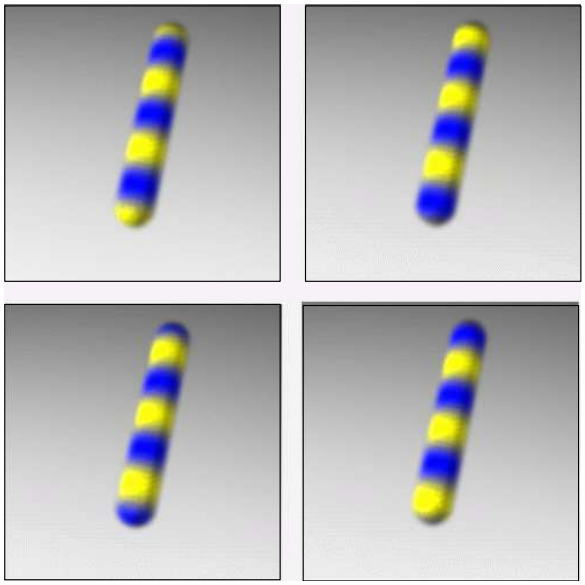

Figure 5: The capsule shape of the flow obtained with $u = 1$ ($C_0 \neq C_1$).

For rendering 3D flow we first sample color and opacity functions, $C(v,t)$ and $O(v,t)$ and obtain a set of volumes with associated color and opacity. We then render these volumes using a ray-tracing version of the Volume Rendering method of Drebin, Carpenter and Hanrahan [**?**]. These volumes can be an object in a ray tracing scene, i.e. they can create shadow and reflection over other objects in the scene. For instance, the scenes in the Figures **??** and **??** include a background plane. Figure **??** shows shadow of the flow over this background plane. The volume object can have specular reflection. For instance, In Figure **??**, the flow has specular reflection in addition to diffuse reflection. Similar to 2D case, to avoid temporal aliasing the functions need to be sampled with a spatial and temporal jittering [**?**].

2.4 Flow Functions for Multiple Lines

If there exists more than one line, the problem is to appropriately combine the functions $p(v,t)$ and $a(v)$ associated with each line. To compute the combined translation, we simply calculate a weighted average of the functions described by each directed line. Let $p_i(v,t)$ and $a_i(v)$ denote the functions described by

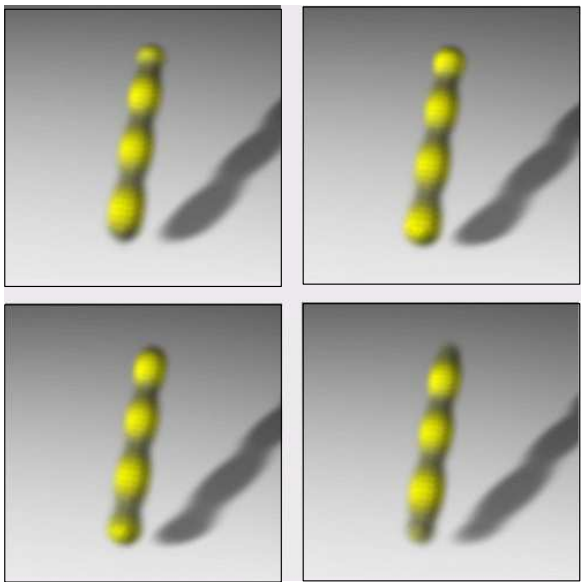

Figure 6: The bloby flow obtained with $u = 0.25$ ($C_0 \neq C_1$).

each line i where $i = 0, 1, \ldots, n$. The combined functions $p(v,t)$ and $a(v)$ are

$$p(v,t) = \frac{\sum_{i=0}^{n} w_i p_i(v,t)}{\sum_{i=0}^{n} w_i}$$

$$a(v) = \frac{\sum_{i=0}^{n} w_i a_i(v)}{\sum_{i=0}^{n} w_i}$$

where w_i is a positive valued function that is computed using the distance from the line i. (The function $f_i(v,t)$ is computed as before, in equation **??**.)

Let $d_i(v)$ be the distance from the line i described earlier. Any monotone decreasing always positive function of d_i can be used as w_i. One example of such functions is

$$w_i(d_i) = exp(-d_i/\mu_i) \tag{4}$$

where μ_i is any positive real number. Larger μ_i values give smoother warping

effects. Beier and Neely used the following function for w_i:

$$w_i(d_i) = \left(\frac{l_i}{a_i + d_i} \right)^{b_i}$$

where l_i, p_i, a_i and b_i are real constants. Beier and Neely suggest that values of b_i in the range of $[0.5, 2]$ are the most useful values. The value of a_i must always be positive. Smaller values provide more precise control while larger values yield smoother warping. The scalar l_i can be used to give different importance to each line. In our examples we generally use simple exponential function given in equation **??**.

An important concern in choosing an approapriate blending operations is their associativity and commutativity [**?**]. If associative and commutative blending operations such as the one given in equation refeq3 are used, the resulting flow functions $F(v, t)$ can be recomputed in constant time (i.e. it is constant time updateable) after deleting, changing or adding a constant number of line segments [**?**].

2.5 Rendering Flow Described by Multiple Lines

Similar to one line case, we first map the function values to opacity and color. The color and opacity functions, $C(v, t)$ and $O(v, t)$, are computed using the same functions given in equations **??** and **??**.

For rendering 2D flow, we again sample color and opacity functions, $C(v, t)$ and $O(v, t)$, and obtain a set of image frames with associated color and opacity. We obtain image frames for 2D flow animation by compositing these images with a background image. As we have mentioned earlier, in order to avoid temporal aliasing the functions need to be sampled with a spatial and temporal jittering [**?**].

We have developed a user interface to create 2D flow animations. Using this interface we have created various 2D flow animations. Figure **??** and **??** show oriented control lines and related frames of the cyclic animations. The bottom four images in Figure **??** and **??** are cyclic animation frames for the given set of oriented line segments that are shown on the top row.

For rendering 3D flow decribed by multiple lines, we again sample color and opacity functions, $C(v, t)$ and $O(v, t)$, and obtain a set of volumes with associated color and opacity. We then volume render these volumes as before [**?**]. Figures **??**, **??**, **?? ??**, **??**,and **??** show cyclic 3D animation frames for two sets of oriented 3D line segments with three different u values. These animations can be seen at *www-viz.tamu.edu/faculty/ergun/research/flow/animations/*.

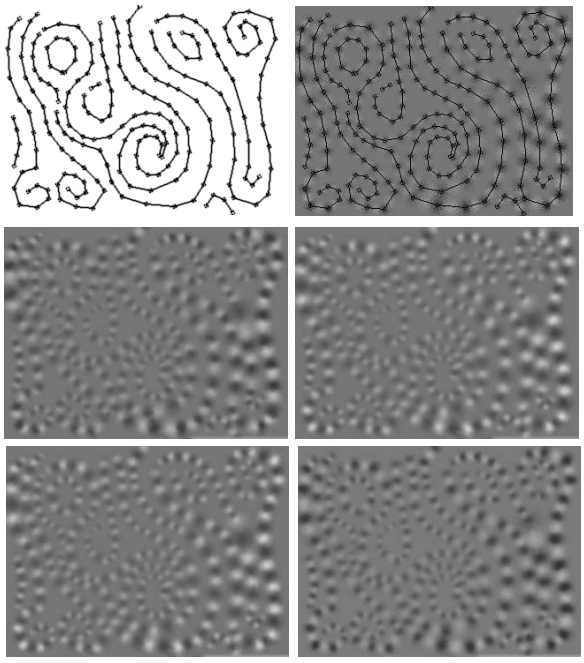

Figure 7: Oriented lines and related frames of cyclic animation.

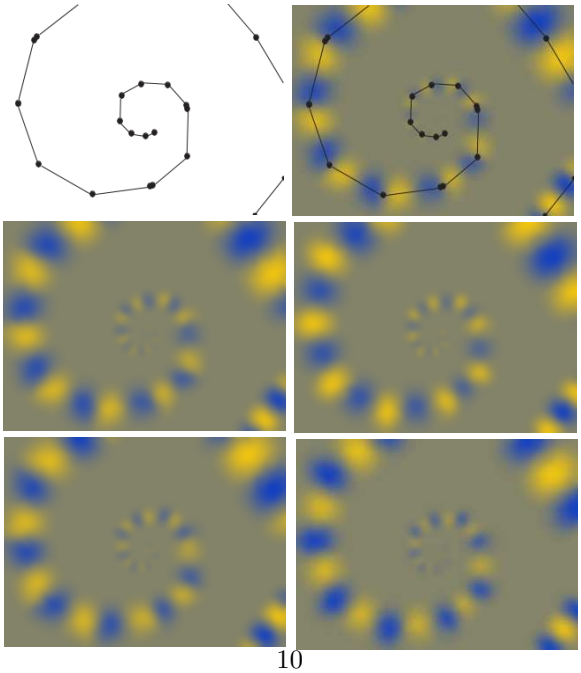

Figure 8: Another example of cyclic animation.

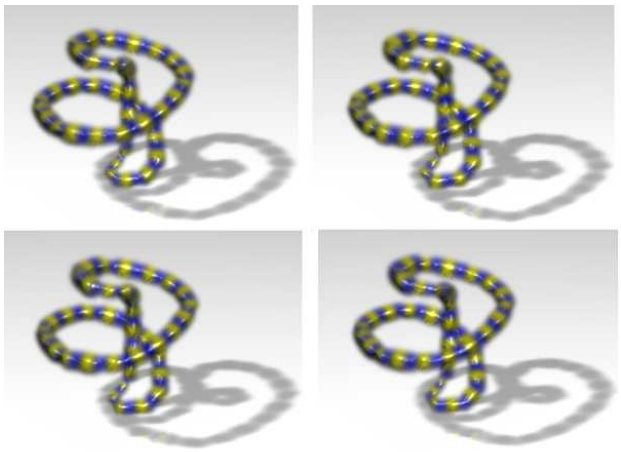

Figure 9: A flow modeled by multiple lines in 3D ($u = 1, C_0 \neq C_1$).

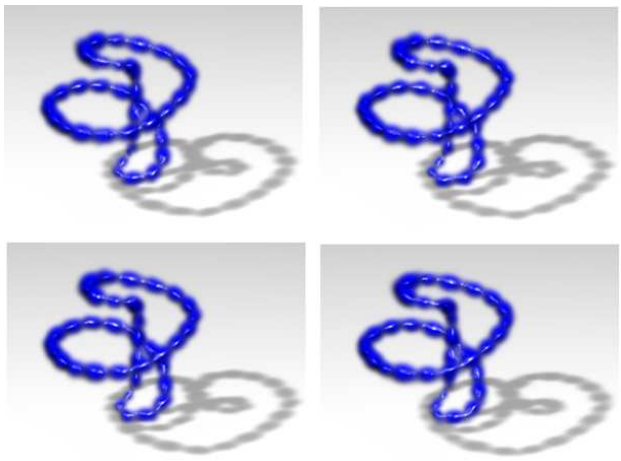

Figure 10: A flow modeled by multiple lines in 3D ($u = 0.25, C_0 = C_1$).

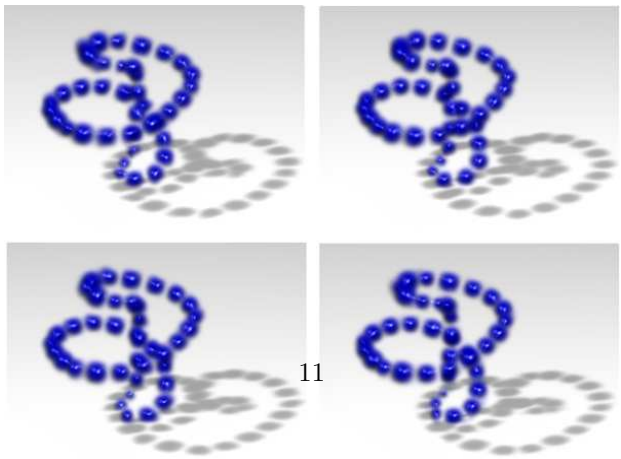

Figure 11: A flow modeled by multiple lines in 3D ($u = 0.0, C_0 = C_1$).

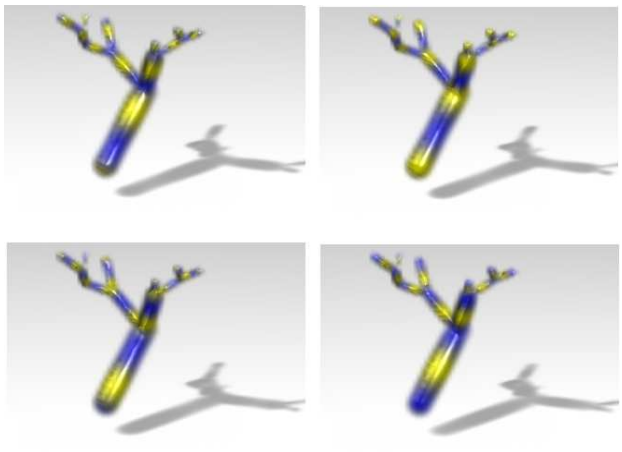

Figure 12: A tree like flow modeled by multiple lines in 3D ($u = 1, C_0 \neq C_1$).

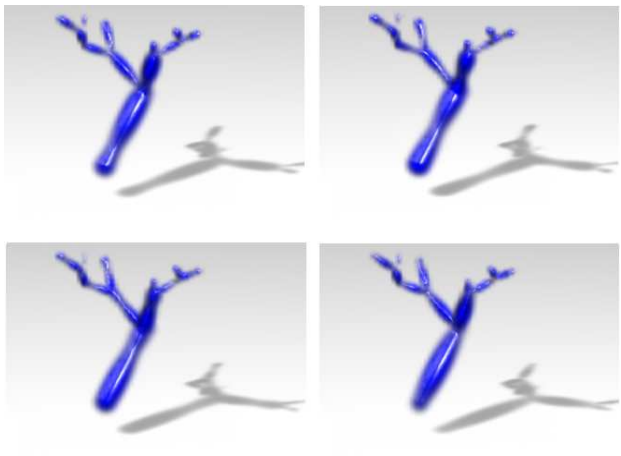

Figure 13: A tree like flow modeled by multiple lines in 3D ($u = 0.25, C_0 = C_1$).

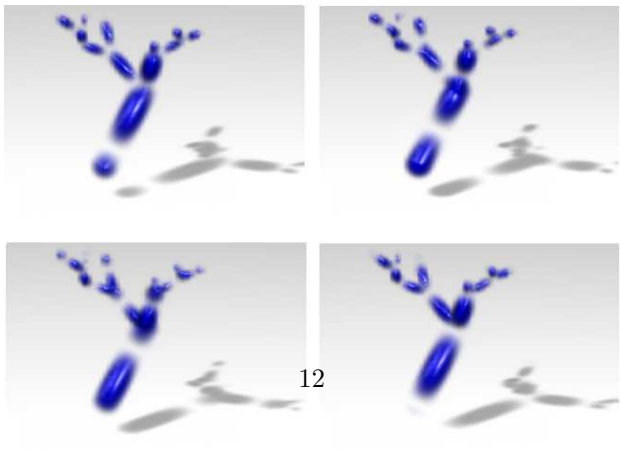

Figure 14: A tree like flow modeled by multiple lines in 3D ($u = 0.0, C_0 = C_1$

3 Application of Flow Animations to Compressor Visualization

We used the 2D flow animations to improve the quality of compressor visualizations. The compressors are typical of the type found in household refrigerators and air-conditioning systems. There are various types of compressors such as centrifugal, scroll, rolling piston and twin-screw [**?**]. For heating, refrigerating and air conditioning engineers it is essential to understand and visualize how each type of compressor works. Therefore, the goal of compressor visualization is to illustrate the compression principles for different types of compressors. Unfortunately, it is almost impossible to understand the working principles of compressors just by looking at static illustrations. It is therefore essential to create animations that show how the individual parts of compressors and the associated fluid move during compression cycles. However, if the animations of the flow were not included, then the viewer is left to imagine how the gases are compressed.

For some compressors, such as scroll, rolling piston and rotary vane, the best staging of the compressor motion is given by parallel projection. For the visualization of such compressors 2D flow animations can be a useful tool. We used 2D flow animations to improve the quality of visualization of rolling piston and rotary vane compressors. Rolling piston compressor uses a roller mounted on the eccentric of a shaft with a single vane or blade suitably positioned in the nonrotating cylindrical housing, generally called the cylinder block. The blade reciprocates in a slot machined in the cylinder block. This reciprocating motion is caused by the eccentrically moving roller. In order to create rolling piston animation, we have developed a simple texture mapped 3D model for the compressor and animated the cylinder blocks by using a commercial modeling and animation package. We have then rendered this animation by using parallel projection. Flow animations are created in our system separately. These two animations are later combined by using a compositing program. Some frames of resulting animation are shown in Figure **??**. The animation of rolling piston can be seen at *www-viz.tamu.edu/ASHRAE/current/cdrom/chapter/section3.html*.

As an example of rotary vane compressors we used an eight-bladed compressor. The eight discrete volumes are referred to as cells. In this compressor, a single shaft rotation produces eight distinct compression strokes. In a similar way to the rolling piston case, a rotary vane compresser is modeled, animated and rendered in 3D by using a commercial modeling and animation package. Flow animations are created in our system separately. Then the two animations are combined Using a commercial compositing package. Some frames of the resulting animation are shown in Figure **??**. The animation of rotary vane can be seen at *www-viz.tamu.edu/ASHRAE/current/cdrom/chapter/section3.01.html*.

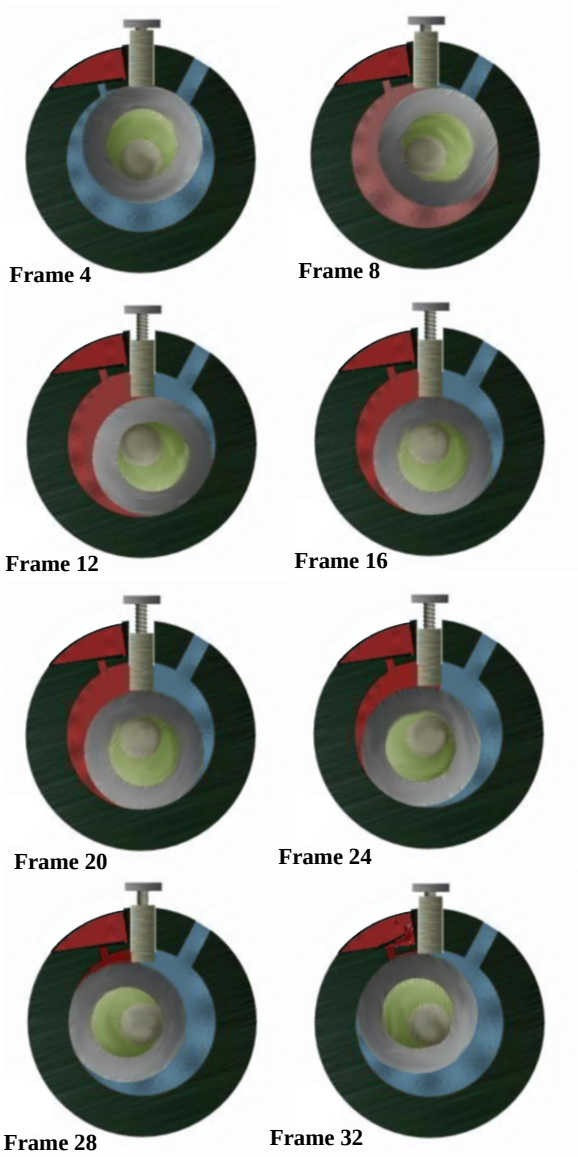

Figure 15: Sample frames of a rolling piston animation.

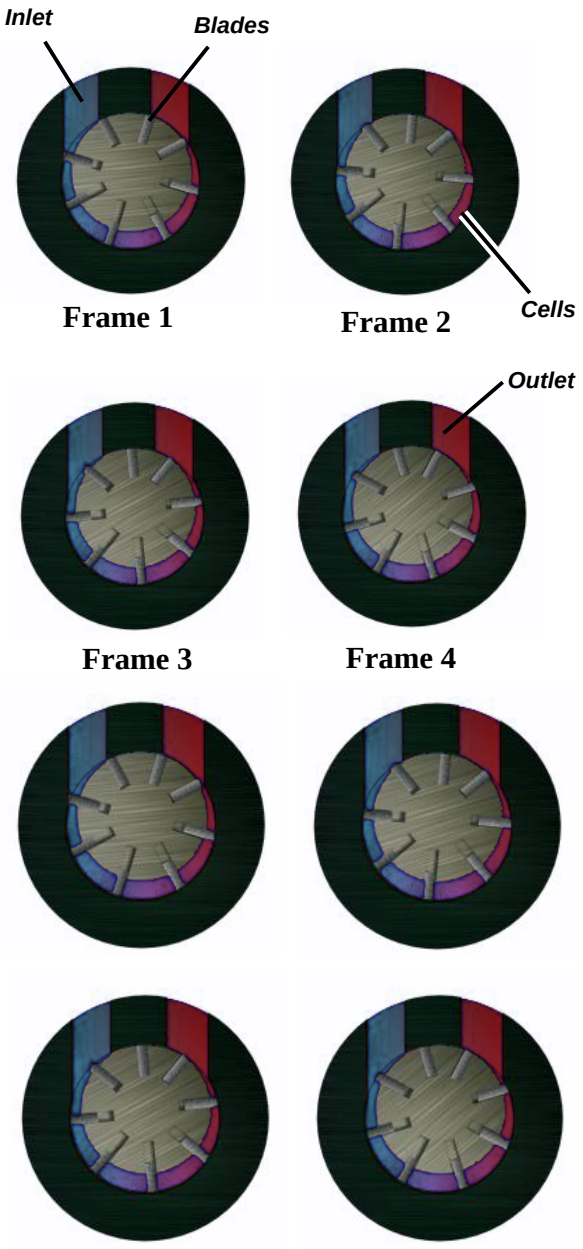

Figure 16: Sample frames of rotary vane animation.

4 Conclusion

In this paper, we have developed a function-based approach for user controlled flow animation. Our approach provides a simple algorithm for modeling variable-speed flow animations in both 2D and 3D.

5 Acknowledgments

Portions of this work were supported by The American Society of Heating Refrigeration and Air Conditioning Engineers (ASHRAE) through research project 1017-RP. Special thanks to Sajan Skaria who developed the models and animations of the rotary vane and rolling piston compressors.

References

[1] B. Cabral and L. Leedom, "Imaging Vector Fields using Line Integral Convolution", *Proceedings of ACM SIGGRAPH'93,* pp. 263-272, August, 1993.

[2] W. T. Freeman, E. H. Adelson and D. J. Heeger, "Motion without Movement", *Proceedings of ACM SIGGRAPH'91,* pp. 27-30, July, 1991.

[3] L. K. Forsell, "Visualizing Flow over Curvilinear Grid Surfaces Using Line Integral Convolution", *Proceedings of IEEE Visualization'94,* pp. 240-247, October, 1994.

[4] L. Forsell and S. D. Cohen, "Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation and Unsteady Flows", *IEEE Transaction on Visualization and Computer Graphics,* vol. 1, no. 2, pp. 133-141, June, 1995.

[5] D. Stalling and H. C. Hege, "Fast and Resolution Independent Line Integral Convolution", *Proceedings of ACM SIGGRAPH'95,* pp. 249-256, August, 1995.

[6] N. Max and B. Becker, "Flow Visualization Using Moving Textures", *Proceedings of ICASE/LaRC Symposium on Visualizing Time-Varying Data,* 1995.

[7] A. Van Gelder ad J. Wilhelms, "Interactive Visualization of Flow Fields", *Proceedings of Workshop on Volume Visualization,* pp. 47-54, 1992.

[8] B. Jobard and W. Lefer, "The Motion Map: Efficient Computation of Steady Flow Animations", *Proceedings of IEEE Visualization'97,* pp. 323-328, October, 1997.

[9] J. Bloomenthal, C. Bajaj, M-P Cani, A. Rockwood, B. Wyvill and G. Wyvill, *Introduction to Implicit Surfaces,* Morgan Kaufmann, 1997.

[10] A. Ricci, "A Constructive Geometry for Computer Graphics", *The Computer Journal,* vol 16, no. 2, pp. 157-160, May 1973.

[11] Beier and Neely. Feature-Based Image Metamorphosis. *Proceedings of ACM SIGGRAPH'92*, 26(2):35-42, 1992.

[12] B. Crespin. Implicit Free-Form Deformations. *Proceedings of Implicit Surfaces'99*, pp. 17-24, September 1999.

[13] G. Wyvill, C. McPheeters and B. Wyvill, "Data Structure for Soft Objects", *The Visual Computer,* vol. 2, no. 4, pp. 227-234, 1987.

[14] J. Blinn, "A Generalization of Algebraic Surface Drawings", *ACM Transactions on Graphics,* vol. 1, no. 3, pp. 235-256, 1982.

[15] R. Cook, T. Porter andL. Carpenter, "Distributed Ray Tracing", *Proceedings of ACM SIGGRAPH'84,* pp. 137-145, August, 1984.

[16] R. A. Drebin, L. Carpenter and P. Hanrahan, "Volume Rendering", *Proceedings of ACM SIGGRAPH'93,* pp. 65-74, August, 1988.

[17] E. Akleman and J. Chen, "Constant Time Updateable Operations", *Proceedings of Shape Modeling'99*, 73-80, September 1999.

[18] *ASHRAE, H.V.A.C. Systems and Equipment: Chapter 34 - Compressors,* American Society of Heating, Refrigerating and Air-Conditioning Engineers, Atlanta, GA., 1996.