# Using Motion to Illustrate Static 3D Shape - Kinetic Visualization

Eric B. Lum Aleksander Stompel Kwan-Liu Ma
University of California, Davis

*Abstract*—

**In this paper we present a novel visualization technique – *kinetic visualization* – that uses motion along a surface to aid in the perception of 3D shape and structure of static objects. The method uses particle systems, with rules such that particles flow over the surface of an object to not only bring out, but also attract attention to information on a shape that might not be readily visible with a conventional rendering method which uses lighting and view changes. Replacing still images with animations in this fashion, we demonstrate with both surface and volumetric models in the accompanying videos that in many cases the resulting visualizations effectively enhance the perception of three-dimensional shape and structure. We also describe how for both types of data a texture-based representation of this motion be rendered using PC graphics hardware for interactive visualization. Finally, the results of a user study that we have conducted is presented, which show evidence that the supplemental motion cues can be helpful.**

*Keywords*— **animation, visual perception, particle systems, scientific visualization, volume rendering, graphics hardware, texture.**

## I. Introduction

Scientific visualization is often concerned with the creation of 2D visual depictions of the features found in 3D data sets with the goal of having these 2D images provide scientists with insights into their data. Unfortunately, this intermediate 2D representation can introduce ambiguities since it is merely a 2D projection of this 3D data. This is particularly a problem when rendering semi-transparent materials with direct volume rendering. In this case, changes in pixel luminance can ambiguously be the result of a number of factors. It can be caused by the illumination of any one of the overlapping surface layers, it can be an indication of opacity, or can be a result of the color map specified by the transfer function. Our work deals with the use of motion as a way of providing supplemental cues to aid in the perception of these often ambiguous 3D structures found in scientific visualization.

Time-varying sequences of images are widely used in visualization as a means to provide an extra dimension of information for perception to occur. This animation might be as simple as the changing of camera or object positions or can include animations resulting from time-varying changes in the data itself. However, using motion that is independent of changes in viewing direction for conveying the shape of *static objects* has been a rather unexplored area. In this paper, we present a visualization technique, which we call *kinetic visualization*, for creating animations that illustrate the shape of a static object in an intuitive manner using motion.

This work is motivated by the observation that the flow of fast moving water over a rock, a dynamic flame from an open fire, or even a school of fish exhibit motion that gives the perception of shape. Our technique, the basic idea of which is also described in our previous work [1], is built on the inspirations we received from kinetic art [2], the studies done in cognitive science, specifically on structure-from-motion perception [3] [4], the ideas of particle systems [5], and the work of Interrante [6] on using texture to convey the shape of overlapping transparent surfaces. It is unique because we are able to apply motion as a supplemental cue to enhance perception of shape and structure, and because the motion is created not only according to the characteristics of the data but also using a set of rules based loosely on physics and biology. A static image from an animation generated using our technique is shown in Figure 1. Because of the nature of the techniques presented, readers are advised to watch the accompanying videos in order to follow the exposition. The videos can be downloaded from:

`http://www.cs.ucdavis.edu/~ma/kinvis/`

The technique introduced in this paper is not meant to be a replacement for traditional rendering techniques that use lighting and viewpoint changes to indicate shape, rather it can augment those methods for more intuitive and effective visualization as illustrated in Figure 2.

We expand on our previous work [1] in this paper, describing how a texture based representation of this motion can be used for improved interactivity with a larger number of motion primitives using commodity PC graphics hardware. In particular we describe how 2D textures can be used to remove the need for sending position information of each motion primitive across the graphics bus. For volume data we describe how by using dynamic volumetric particle textures with the multi-texturing

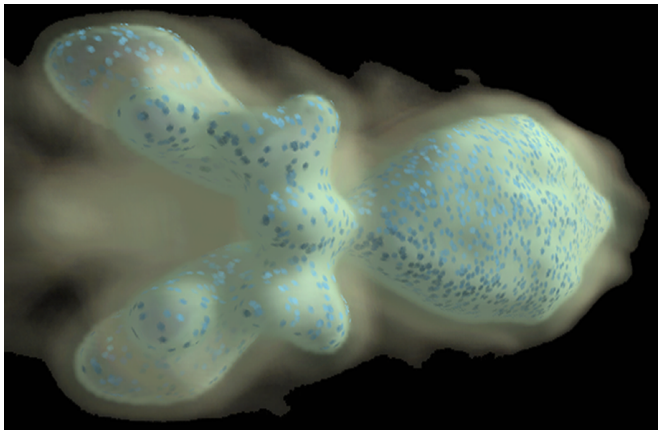


Fig. 1. A single frame of an animation showing a PET scan of a mouse brain. The particles help to illustrate one of the function levels while direct volume rendering gives context to their motion. The method described in this paper uses the motion of particles to illustrate shape. As such, a static image like the one shown does not demonstrate the technique. The reader is encouraged to watch the accompanying videos.

* CIPIC & Department of Computer Science, University of California, One Shields Avenue, Davis 95616, {lume,stompel,ma}@cs.ucdavis.edu

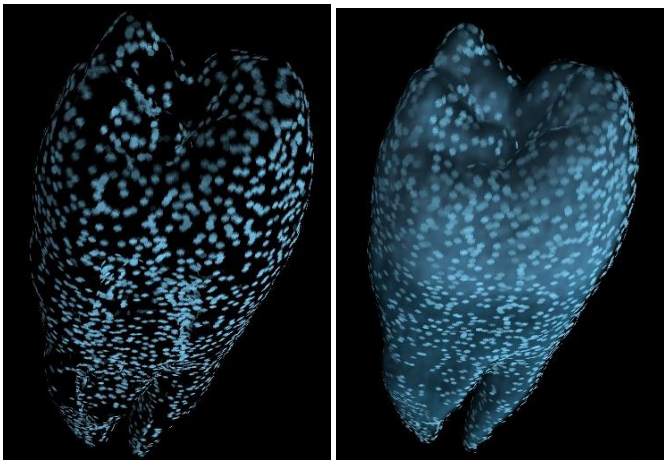


Fig. 2. The kinetic visualization technique we describe is not meant to be a replacement for conventional rendering methods. For example, the moving particles on the left are used in combination with traditional volume rendering to create the visualization on the right. Since our technique uses motion to illustrate shape, neither of these still image is representative of our technique.

capability of commodity PC graphics cards and the temporal compression technique described in [7] we achieve interactive texture-based volumetric kinetic visualization. It is important to point out that our technique is based on motion, rather than texture, to illustrate shape; graphics texture hardware is merely used as a means of generating that motion.

## II. Visual Cues

In recent years there has been increasing amounts of research in the application of non-photorealistic rendering (NPR) to scientific visualization with the goal of creating more perceptually effective visualizations [8] [6] [9]. The reason being that artists for centuries have dealt with the situation of trying to create meaningful representations of a 3D world using 2D media. Although NPR techniques are often beneficial, the modifications made by these techniques to the hue and luminance of the pixels in an image can also introduce their own type of ambiguity. As an example, aerial perspective [10] might be used to illustrate depth, where far away objects are rendered in cooler blue less saturated colors, similar to how an artist might illustrate distant mountains. This can introduce ambiguity with respect to the use of the color blue, since it can become unclear if an object is blue because of its material or because of its depth.

To present more shape cues to the viewer, our technique uses motion as a means of providing supplemental information to the user. This does, however, also introduce its own type of ambiguity. Namely, since our method uses motion to illustrate shape, it can also give the impression of motion or direction that is not physically present. Therefore, the technique is not appropriate for the visualization of time-varying phenomena since the motion used could give the impression of changes not present in reality. Our technique is also not appropriate for vector-field data sets since the particle movement can convey direction contrary to that found in the data. Nevertheless we believe there is a large class of static scalar data sets to which the method is applicable.

We have applied kinetic visualization to two different types of static data. One includes surface models represented as polygonal meshes, in which case particle motion is influenced by surface normal, principal curvature direction, and curvature magnitude. The other type of static data is regularly sampled scalar volumetric data, where scalar value, gradient magnitude, gradient direction, principal curvature direction, and transfer function are used in the calculation of particles motion.

## III. Related Work

The perception of shape through motion, called "structure-from-motion", has long been studied in psychology [11]. Treue et al. [4] demonstrate that the movement of points on an object can give the perception of shape, using as stimulus a rotating cylinder with a random dot pattern on the surface. Their work shows a "building up" time is required for mental generation of a surface representation from the integration of point velocities. They also find that subjects were able to perform various tasks with peak performance when points had lifetimes of at least 125 milliseconds (ms), and that with lifetimes of less than 60 ms shape perception from motion did not occur. It should be noted, however, that the dot patterns in their research were attached to a rigid structure rather than moving over the structure itself as is the case with our work.

Further work by Andersen and Bradley [3] demonstrates that structure-from-motion perception requires either a large number of dots, or fewer dots that appear in varying positions over time. Their work also suggests that the middle temporal area (MT) of the brain is essential for the structure-from-motion perception.

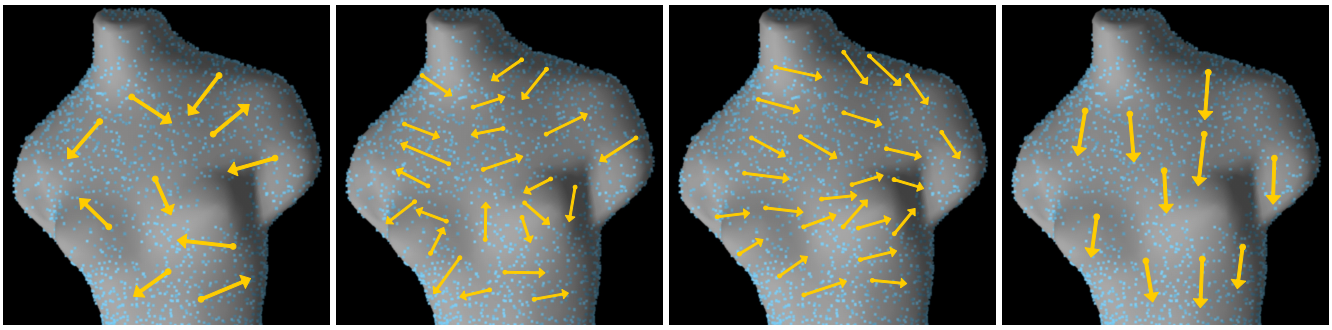
Wanger, Ferwerda, and Greenberg [12] explore visual cues by conducting three psychophysical experiments in which the accuracy of interactive spatial manipulation performed by subjects was measured. Their study shows that different visual cues facilitate different tasks. Motion is found to have a substantial positive effect on performance accuracy of orienting tasks in which spatial location is less important but relative alignment information is needed. Limoges et al. [13] study the use of motion to give the perception of correlations between variables. Their work focuses on the display of statistical data, not the geometric data that we deal with in our work.

Kinetic art incorporates real or apparent movement in a painting or sculpture. Kinetic artists often use various means to deemphasize form and color in favor of movement. From studies in neurology, it is evident that an entire area of the brain is devoted to processing motion information. Zeki [2] proposes that the same area is essential for appreciating kinetic art.

Motion blur [14],[15] captures the effect of motion in still images and is widely used in producing realistic animations. The work presented in this paper deals with the inverse problem, where instead of using a static image to represent a dynamic phenomenon, dynamic animations are generated for the visualization of static data.

Motion without movement [16] assigns perceptual motion to objects that remain in fixed positions by using oriented filtering. This technique can generate a continuous display of instantaneous motion. Line integral convolution [17], based on the same principle, low-pass filters a noise function along a vector field direction to create visualization of direction information.

Our work applies particle systems, which have been used to model a set of objects over time using a set of rules [5]. They



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

Fig. 3. The yellow arrows illustrate the direction of the particles and are not part of our kinetic visualization technique. (a) Particles are constrained to lie along an object's surface. Without the addition of other rules, particles simply move with random trajectories along the surface. (b) Particles flow over an object following the first principal curvature directions(PCD). Following the first PCDs does not guarantee that particles follow an object's maximum curvature with consistent directions. (c) The use of flocking adds local consistency to the particle directions. By limiting the degree flocking is applied, particles are still influenced by the first principal curvature directions. (d) Particle directions can also be made more consistent by making the particles have a tendency to move in a user specified "preferred" direction resulting in an appearance that in some ways resembles the flow of water over an object.

have been applied to the modeling of a wide variety of phenomena, including smoke, fire, and trees, using a set of either deterministic or stochastic rules of motion [10]. These rules can be based on physics, for example gravity, or even biology, as is the case with flocking behaviors.

The shape, density, transparency, and size of particles can have an impact on the visual appearance and resulting perception cues. Interrante [18] has done a comprehensive study on using opaque stroke texture to improve the perception of shape and depth information of overlapping transparent surfaces. Our work considers particle shape to some extent, but the focus of our work is particle motion, rather than shape.

Using particles as a representation of shape is also related to point-based rendering. Point-based rendering algorithms typically use reconstruction filters that disguise the appearance of the point representation [19]. In some ways our work can be thought of as a variation of point-based rendering where the points move over time and are intentionally made visible.

In the volumetric case, our work is analogous to splatting [20] with a limited budget of splats. The location and size of each particle are not specified to represent the entire volume, but rather are positioned such that their location and movement create a dynamic representation of the static volume. In this way, our technique allows for the volume visualization of extremely large volumetric data sets with a limited rendering budget.

## IV. MOTION STRATEGIES

In this section we discuss the set of rules we apply to generate geometrically meaningful motion. The overall goal is to create rules resulting in particles that indicate shape by smoothly flowing over an object, with locally consistent directions, and a density distribution that does not let particles "clump" together in regions of little interest. Many of the rules imposed on the particles are loosely based on biology or physics. It is our belief that these types of rules are desirable since they are similar to the types of stimulus the human visual system has been adapted to process.

### A. Motion Along the Surfaces

Since we would like to better illustrate an object's shape, rules are imposed to constrain the motion of particles to a surface. The motion of particles along an object's surface over time presents the viewer with a set of vectors (trajectories) that run parallel to a surface. In the case of viewing a mesh, this rule is accomplished by simply constraining the particles to lie on the mesh as shown in Figure 3(a).

In the case of volumetric data, the rules are applied to restrict motion along the local gradient. Particle movement is reduced along the gradient direction as illustrated in Figure 4 and can be described in the following equation:

$$\vec{D}_{n+1} = \vec{D}_n - (\vec{G} \cdot \vec{D}_n)\vec{G}$$

where $\vec{D}_{n+1}$ is the new particle direction, $\vec{D}_n$ is the particle direction in the previous iteration, and $\vec{G}$ is the gradient direction. This results in particles that move along and do not leave the surface of interest. After every iteration, velocities are normalized to have constant magnitude or speed in 3D world space. A particle with reduced speed in projected screen space thus provides cues that it is either moving in a direction near parallel to the view direction or is far from the viewer and thus has reduced speed on the screen as a result of perspective. If particle speed was allowed to vary, such depth and orientation cues would be lost.

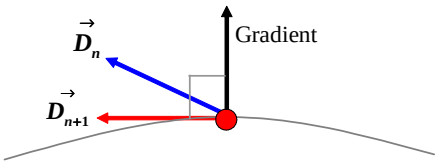



Fig. 4. Particles are constrained to have a direction perpendicular to the gradient. The particle is shown in red, $\vec{D}_{n+1}$ is the new particle direction, and $\vec{D}_n$ is the particle direction in the previous iteration.

### B. Principal Curvature Direction

The principal curvature directions (PCDs) indicate the directions of minimum and maximum curvature on a surface. Interrante [6] describes how line integral convolution along the principal curvature directions can generate brush-like textures that create perceptually intuitive visualizations since the resulting textures "follow the shape" of the object being rendered. Similarly we use principal curvature directions to create particles that "follow the shape" of a surface. Particle directions are adjusted so the particles flow in a manner that favors the first principal curvature direction. This is shown in Figure 5 and can be expressed as:

$$\vec{D}_{n+1} = (1 - k_c c_m)\vec{D}_n + (k_c c_m)\vec{C}$$

where $k_c$ is a scale factor used to specify the degree the principal curvature direction influences particle direction, $\vec{C}$ is the principal curvature direction vector, and $c_m$ is the magnitude of the principal curvature direction vector. Note that a curvature direction at any point is the same forward as backward. When PCD is incorporated with particle direction, its orientation is adjusted so that it is most consistent with the current direction of the particle. The PCD rule results in particles that smoothly flow over an object, although the particles are not guaranteed to move in the same direction as shown in Figure 3(b).

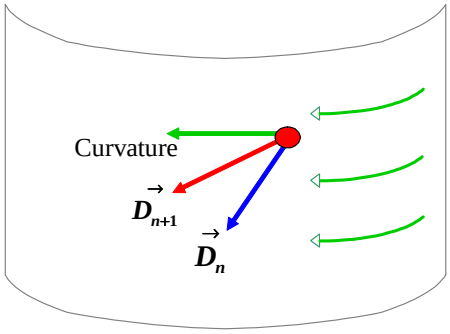



Fig. 5. Particle direction is adjusted to be more closely aligned with the principal curvature direction

### C. Consistent Directions

The motion of dots in opposite directions can suppress response to the middle temporal (MT) area of the brain and can give perceptual cues of differences in depth [3]. We therefore use a set of rules that move particles in directions consistent with their neighbors. This is particularly important since the PCD-based rule in the previous section can cause particles to follow a PCDs in opposite directions. We use two different types of rules to enforce consistency.

The first method we use to give the particles more consistent directions is to assign the particles flock-like behavior. A flocks exhibits motion that is fluid, with each member still exhibiting individual behavior. Thus flocking can be used to add local uniformity to particle motion while still allowing particles to have motion shaped by outside forces like principal curvature direction. Reynolds [21] presents a method for creating flock-like particle systems using behaviors that include velocity matching, collision avoidance, and flock centering. We have found that adjusting particle direction towards that of the flock to be an effective method in yielding more consistent particle directions. This rule makes each particle attempt to match the direction of its neighbors. Flock direction for each particle is calculated as the average of the neighboring particle directions weighted by distance. The manner in which particle directions are adjusted based on flocking is illustrated in Figure 6 and can be expressed by the following equation:

$$\vec{D}_{n+1} = (1 - k_f)\vec{D}_n + k_f\vec{F}$$

where $k_f$ is a constant interactively specified by the user that controls the extent the flock vector affects the particle direction, and $\vec{F}$ is the flock vector for this given particle. At times flocking can result in motion that contradicts other rules that are imposed. By varying the constant $k_f$ and not enforcing strict direction matching, particles can still exhibit motion influenced by other rules, like those involving principal curvature directions, while still adding consistency with respect to their neighbors as shown in Figure 3(c). Collision avoidance is used to give particles a more uniform distribution and will be discussed in the next section. Flock centering is not used since it is not our intention to have the particles stay together as a coherent unit but rather to create particles exhibiting locally flock-like behavior.

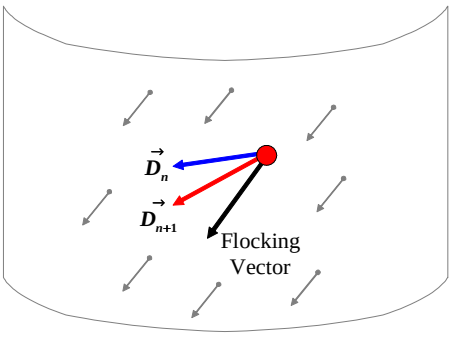



Fig. 6. Using fbcking, each particle has its direction adjusted to be similar to its neighbors' directions.

A simpler method for giving particles a consistent directions is to simply define a "preferred" direction the particles move, as shown in Figure 3(d). This can be expressed by the following equation:

$$\vec{D}_{n+1} = (1 - k_p)\vec{D}_n + k_p\vec{P}$$

and is illustrated in Figure 7 where $k_p$ is a percentage of the contribution by preferred direction and $\vec{P}$ is the preferred direction. The result is a flow of particles that move over a surface with an appearance similar to water flowing over an object. One drawback of this approach is that at the extreme ends of an object, where the particles flow from and flow into, the directions of the particles are not consistent; that is, the particles would move in opposite directions either to or from a point on the surface.

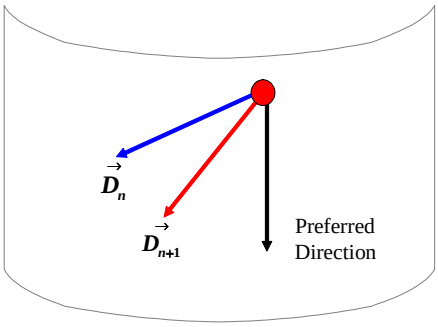



Fig. 7. A particle direction becomes a weighted sum of the previous direction and a user specified "preferred" direction.

### D. Particle Density

Treue et al. [4] demonstrate that if moving stimuli become too sparse, shape perception is diminished. Consideration must therefore be taken with regard to particle density. Since the number of particles has a direct influence on rendering time, it is desirable to have a set of rules that efficiently uses a limited budget of particles. In addition, rules regarding particle density are necessary since following principle curvature directions can result in particles accumulating in local minima.

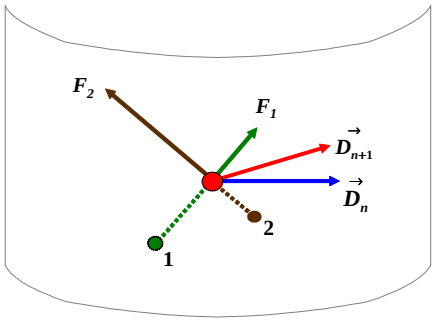



Fig. 8. Particle density is controlled by using magnetic repulsion. All particles have the same charge and are repelled by other particles with forces inversely proportional to the square of their distances. In this case particle one (in green) exerts force $\vec{F}_1$ and particle two (in black) exerts force $\vec{F}_2$ on the red particle in the middle.

By using a set of rules based on magnetic repulsion, more uniform particle densities can be achieved. Particles are modeled as having magnetic charges of the same sign, and are repelled from their neighbors with forces inversely proportional to the square of their distances as shown in Figure 8. This is similar to the rule Reynolds uses for flock collision avoidance [21]. In order to avoid numeric instability from particles that are too close, the total force is clamped. Using this technique yields more uniform particle densities. Use of this rule, however, must be limited, since it can make particles move with directions contrary to other rules.

Another method for controlling particle density is to use particle lifetimes designed to prune particles from high density regions, to be respawned in regions of lower density. During each update iteration, the density around each particle is calculated and the particle is removed with a probability proportional to its density. The calculated density can be artificially manipulated based on other factors such as visibility and curvature magnitude to further prune particles for even more effective use. Removed particles are added to regions that had the lowest densities during the previous iteration. Finally, for volumetric data sets, in order to permit the user to apply the particles to particular structures of interest, a separate particle transfer function can be used to limit the particle to specific ranges of scalar values. Figure 9 shows the result of setting transfer functions to illustrate two different isovalues in a volumetric dataset.

### E. Particle Color

The color of each particle can be varied to provide additional information. Gooch et al. [8] describe how variation in hue from warm to cool can be used to indicate shading, reserving variation in color intensity for outlines and highlights. Schussman et al. [22] use hue to indicate line direction when visualizing magnetic field lines. Either of these ideas can be incorporated as a particle system rule. The hue of each particle can be varied from cool to warm based on lighting. Particles can also have their color temperature varied depending on direction, with particles moving in a direction toward the viewer being rendered in warmer colors than receding particles. Particle color can be used to indicate other scalar values, such as curvature magnitude or gradient magnitude for volumetric data sets.

Special consideration with regard to particle color must be taken into account when the particles are combined with traditional rendering techniques. For example, if particles are to be drawn on top of a surface, the particles should not have a color too similar to the surface or they will not be visible. In addition it is often desirable to have particle color intensity vary based on shading parameters. This is particularly helpful when the particles are dense, since they can obscure the lighting cues provided by the underling surface. If particles are lit, a different set of lighting parameters should be used for the particles in order to avoid their blending in with the surface and becoming difficult to see, especially when a particle is in a darker region. For example, if the particles and surface are both rendered in extremely dark colors, it can be difficult to see the particles, even if they differ in hue from the surface. In our implementation we allow the user to vary the particle color based on gradient magnitude, view vector, and direction vector. Each of these adjustments in color can be used alone or combined together.

To avoid rapid changes in particle color and facilitate the tracking of particles by the user, new particle colors are averaged with their colors from the previous iteration.

### F. Particle Size and Shape

The size and shape of each particle can also influence how it is perceived. For example, if particle size is varied based on density such that the gaps between particles are filled, more traditional point based rendering occurs. Since for our work individual particles must be visible for their motion to be perceived, particles are rendered small enough that the amount of overlap with neighboring particles is minimal. Figure 10 shows four examples of particles rendered in differing sizes.

There are a number of ways that particle size can be varied. Particles can be rendered in perspective such that closer parti-

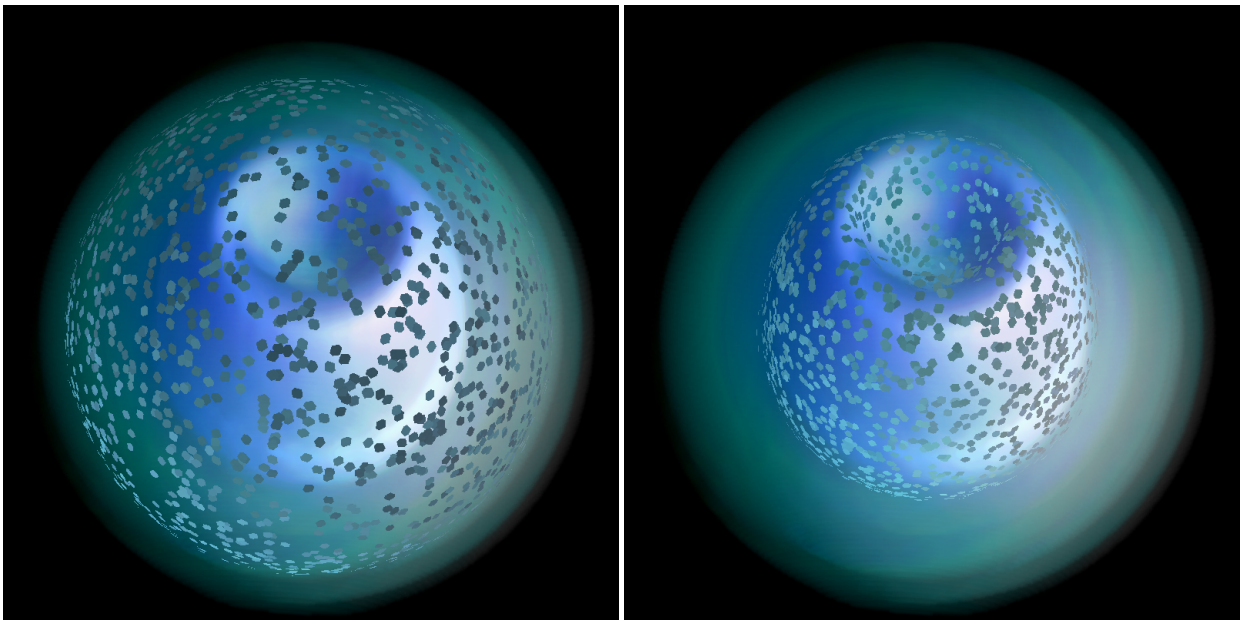


Fig. 9. By using a separate particle opacity map which limits paticles to certain ranges of scalar values in a volume, different function levels can be emphasized and clarifi ed as shown in this two-body electron probability data set.

cles appear larger than further particles, providing a visual cue of particle position as shown in Figure 11. Particle size can be varied based on local density such that the gaps between particles is uniform, similar to splatting. Finally, particle size can simply be kept constant.

Interrante [18] found stroke length to be critically important in her work using strokes oriented along principle curvature directions. Since the emphasis of our work is motion and indicating direction using temporal means, we did not thoroughly investigate how particle shape can be varied to better illustrate shape when combined with motion. As an option, however, particles can be drawn as disc that are slanted to be perpendicular to the surface direction to provide a cue with respect to surface orientation as shown in Figure 11. Particles can also be rendered with a motion blurred stroke-like appearance as a temporal anti-aliasing mechanism.

### *G. Particle Rendering Performance*

We experimentally studied kinetic visualization on a PC with an AMD Athlon 1.4 GHz processor and Nvidia Geforce 3 graphics card. Several of the rules use operations that require access to neighboring particles. For polygon surface rendering, particles are stored in bins on a per polygon basis with the closest particles found by iteratively traversing adjacent polygons. In this manner we are able to render approximately 11,000 particles at 20 frames-per-second using all the rules described in the previous section. For volumetric data, particles are stored in bins determined by hashing a particles spatial location. This is less efficient than binning based on polygons, so we can render 1,500 particles at 20 frames per second. If more particles are used than can be calculated and rendered at interactive rates, animation can be generated in an offline batch mode. Alternately, particle positions over time can also be precomputed in a batch mode process for higher performance rendering, although rendering time still remains proportional to the number of particles since each particle must still be transfered to and rendered by the graphics card.

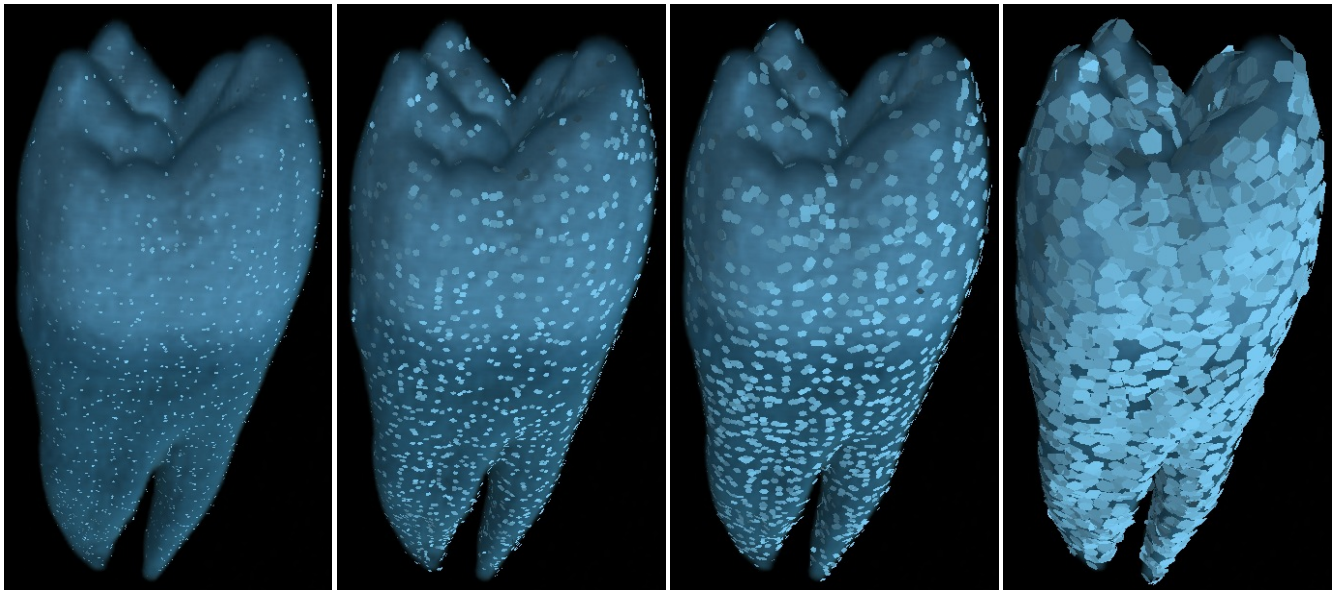
In our implementation the user can turn off and on the various rules and tune motion parameters until the desired visualization is achieved. The interactivity of this process allows the quick selection of parameters that are appropriate for emphasizing the specific regions of interest to the user.

## V. TEXTURE-BASED RENDERING

As an alternative to the point-based rendering of particle motion, we have also investigated the use of dynamic textures for realizing kinetic visualizations. This gives the benefits typically associated with the use of textures, namely the replacement of complex geometry information (per particle positions) with regularly sampled arrays of texels. For polygon meshes 2D textures that vary over time are applied to a surface, while in the volumetric case kinetic visualization textures are used which are blended with direct volume rendering using hardware accelerated, multi-pass, multi-texture rendering techniques.

### *A. Polygon Surface Textures*

Typically the rendering of particles requires the transfer of their positions from main memory to the graphics card across the relatively slow AGP graphics bus for every frame. The amount of data that needs to be transferred is proportional to the number of particles and thus can become a bottleneck in the rendering process when a large number of particles are used. However, by creating time varying 2D textures to which the particles have been applied it becomes necessary to only send this new texture across the graphics bus rather than geometry. Thus by using textures there is a fixed amount of data that needs to transferred for every frame regardless of the number of particles, which for a sufficiently large number of particles yields



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

Fig. 10. Particles are shown rendered in different sizes. Notice that when the particles are rendered too small as shown in (a) they become diffi cult to see. When the particles are rendered too large as shown in (d) it is also diffi cult to resolve the individual particles.

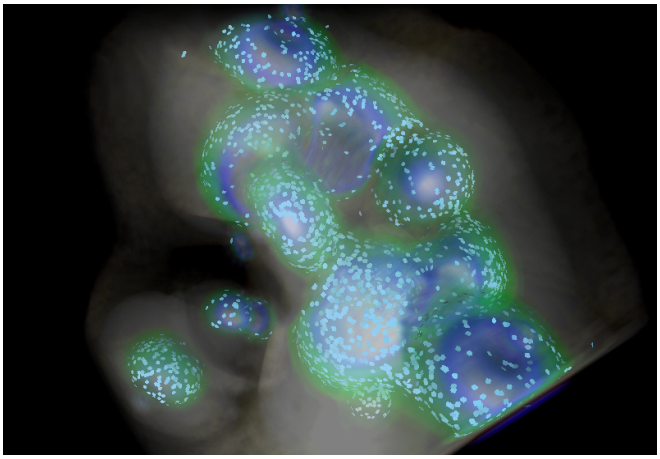


Fig. 11. Particles can be rendered as flat discs that lie perpendicularly to the gradient direction to further indicate surface orientation. The particles shown are also drawn in perspective where farther particles are smaller than nearer particles in this electron distribution simulation volume for a protein molecule.

higher frame rates.

We therefore allow the user to generate a set of video textures that are precalculated in a batch mode process and played back during rendering. Unlike when a normal 2D video, during playback the video textures permit the user to interactively rotate a surface while it is rendered in real time. The main advantage of this technique is that it eliminates the need for a high speed CPU for computing particle trajectories and only requires a PC with 2D texture hardware to work. Since the kinetic visualization textures are precomputed in a batch process, a larger number of particles can be used and more computationally expensive rules for controlling particle direction can be applied. For example when computing flocking behavior or magnetic repulsion based density control, a larger neighborhood of particles can be used for making these calculations.

It is desirable to use shorter animation sequences so the texture data fits entirely in video memory, or at least entirely in main memory. For example, rendering the venus model with a short sequence of time varying $2048 \times 2048$ textures that fit entirely in video memory can be rendered at 300 frames per second with a Geforce 3 since no data needs to be transfered across the AGP bus. On the other hand, rendering occurs at 17 frames per second if the textures must be transfered across the AGP bus from main memory. However, if the animated textures are played in a loop, discontinuities appear between the first and last time step which can be particularly distracting when fewer time steps are used. To avoid this discontinuity particles should therefore have temporal overlap between looped sequences. To create a looped video sequence of length T, particles are limited to having a life-time less than T and are initialized with a randomized starting times. The looped particles sequence is taken from some simulated windowed sequence of length T, with all particles initialized prior to the start of that window removed, and replaced with particles at the end of the windowed sequence as shown in Figure 12. Although this does not strictly enforce density and flocking behavior, in practice, we have found this to severely reduce discontinuities that exist between windowed animation sequences.

It is desirable to allow the user to dynamically vary the particle density during playback. This can be accomplished using a variation of the tonal art maps texturing technique described by Praun et al. [23]. Multiple sets of kinetic visualization textures can be created with differing levels of particle density, with each lower density texture containing a subset of the particles in the previous level. During rendering the two texture levels closest in density to the desired density are combined using multitexturing with blending to approximate the desired particle density. The main disadvantage of this technique is the extra storage requirement of having the different texture levels. If the textures

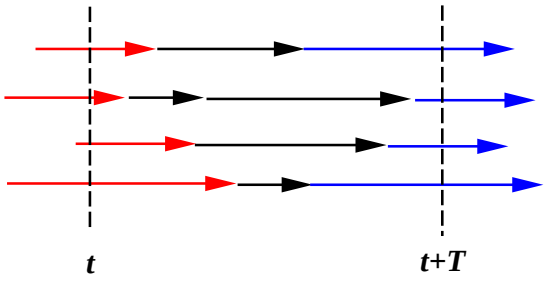



Fig. 12. In order to create looped animations of length T, those particles that exist prior to some time t (shown in red on left) are removed, and replaced with those particles that exist at time t+T (shown in blue on right).

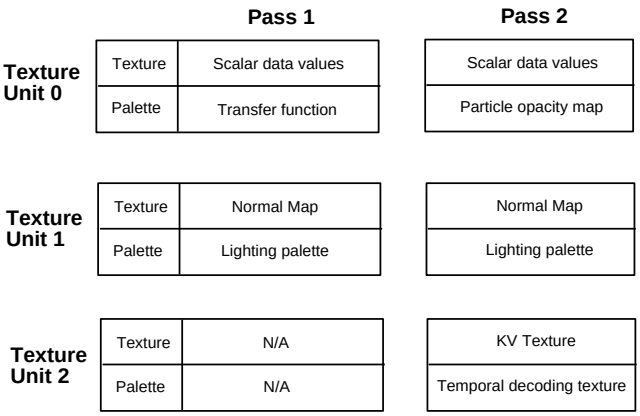



Fig. 13. Rendering requires two passes. The first applies traditional direct volume rendering while the second rendering pass applies the motion textures.

are stored in main memory, it also requires the transfer of two textures per time step rather than one across the graphics bus.

### *B. Volumetric Textures*

For visualizing volumetric data sets using kinetic visualization texture, time-varying volumes of particle textures are used. The time-varying volumes are pre-computed and consist of particles that have been simulated for the entire volume (all scalar values). During final display, particles are rendered only for those scalar values of interest using a separate transfer function for the particles. This approach has the advantage of allowing for the offline simulation of a much larger number of particles than could be simulated at interactive rates. In order to create volumetric video loops of the textures, the same technique for creating cyclical texture movements as described in the previous section can be applied.

Volume rendering of this data is accomplished by using texture hardware [24] and requires two rendering passes with multitexturing that uses three texture units as seen in Figure 13. The first rendering pass performs traditional direct volume rendering with the scalar data and transfer function palette in the first texture and lighting using a paletted normal-map in the second unit. The second rendering pass applies the particles to the volume. The palette in the first texture unit is a transfer function that determines for which scalar values in the volume particles are rendered. This is blended with the lighting in the second texture unit and finally applied to the particles in the third unit.

One limitation of a volumetric representation of the particles is that the time-varying textures can be extremely large. If the volumes have a resolution of 256 in each dimension, for example, each time step requires 16 megabytes of data, making it difficult to fit more than a couple of time steps in video memory, or even tens of time steps in main memory. If these time varying volumes are stored in main memory, the I/O requirements of sending each time step to the graphics card for every frame can severely hamper performance. We therefore apply the temporal compression technique described by Lum et al. [7] to compress these textures by up to a factor of eight. This permits significantly more time steps to be used, and yields interactive frame rates since less texture data needs to be sent from main memory to the graphics card for each frame. For example, using a Athlon 1.2 Ghz PC with a Geforce 3 Ti 200 graphics card, a $256\times256\times256$ volume can be at approximately 3 frames per second without compression, but 6 frames per second using compression by a factor of four.

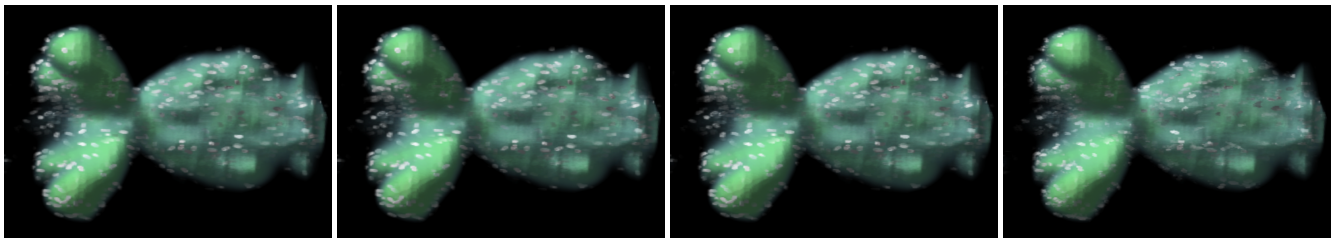
Since this lossy compression technique relies on temporal coherence to reduce the size of the textures, we have found it to work well under conditions that particles move slow enough that some frame to frame texel overlap of the particles exists. This is typically the case since the desired motion textures should have temporal coherence between frames so the particles can be tracked by the viewer over time. Figure 14 shows a rendered volume with particles that have been compressed using varying levels of compression. Image quality is reasonable up to compression levels of four, and falls off noticably at a compression level of eight.

Tradeoffs exist between the use of particle and texture-based representations of kinetic visualization motion. A texture-based prepresentation moves the calculation of particle trajectories to a pre-computation step allowing for the rendering of more particles than could be computed in real-time. The texture based representation, however, does not allow users to have the same flexibility in varying particle properties such as the influence of flocking and magnetic repulsion. The pre-computation step can also take several minutes to complete, especially when using an extremely large number of particles. It is our belief that the point-based rendering of particles has advatanges for visual exploration of a data set, while a texture-based representation is more suited for the presentation of these structures once they have been found.

## VI. DEMONSTRATION

To demonstrate kinetic visualization, several animation sequences have been made and included in videos accompanying this paper and can be downloaded at the URL given previously. Note that all rendering, including volume rendering, was done in hardware to achieve maximum interactivity. Consequently, the image quality, especially for the volumetric models, is not of the same quality as what a software renderer could achieve. The four models used consist of the following:

- A PET scan of a mouse brain ($256\times256\times47$)
- A fuel injection simulation($64\times64\times64$)
- An electron probablitiy distribution simulation($64\times64\times64$)
- A CT tooth volume ($256\times256\times161$)
- A distorted sphere model (15872 polygons)



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

Fig. 14. This images of a PET scan of a mouse brain has been rendered using a texture-based representation of the particle motion with differing level of lossy compression. There is very little noticeable quality degradation until compression by a factor of eight. (a) Uncompressed (b) 2x compression (c) 4x compression (d) 8x compression

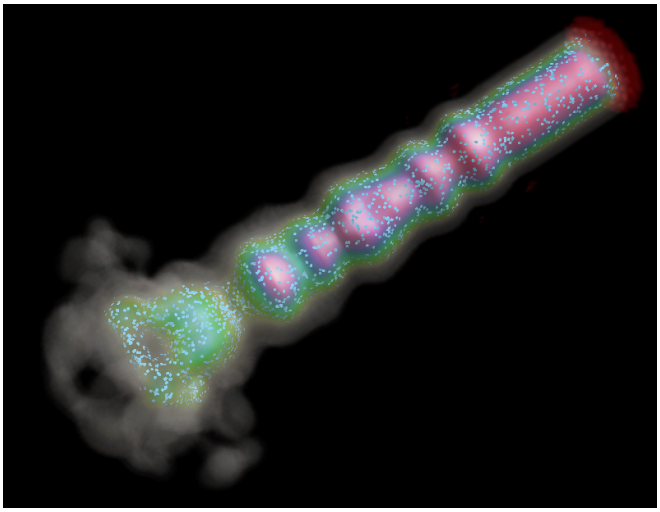


Fig. 15. A single frame of an animation showing a simulation of fuel injection into a combustion chamber.

- A subdivided Venus model (5672 polygons)

The video sequences (`mouse1.mpg`) , (`fuel.mpg`) , and (`neghip.mpg`) shows the use of our technique in the visualization of volumetric data sets. The particles help to illustrate one of the function levels while direct volume rendering gives context to their motion. A single frame from (`fuel.mpg`) is shown in Figure 15. The video sequence (`comparison.mpg`) begins with a still image that shows the type of shape ambiguity that can exist with traditional rendering techniques. With the addition of the particles, the shape becomes immediately apparent. It is not the particles by themselves that clarify the shape, rather, it is the extra shape cues they provide that work in addition to traditional rendering.

The "rules" videos give examples of each of the different rules we apply. Notice in `pcd.mpg` that with the absence of rules, the random motion of the particles on the Venus model does little to clarify shape. By having the particles follow the first principal curvature direction, the particles clearly "follow the shape" of the model.

The next sequence (`flocking.mpg`) shows particles moving along the tooth data set, but with locally inconsistent directions. Although the particles seem to have a slight shape clarifying effect, their contrary motions are distracting and make them difficult to follow. With the addition of flocking, the particles still move along the shape of the tooth, but move in a much more locally consistent manner. In the following sequence, particles flow down the Venus model, in a manner similar to water. The downward tendency adds consistency to the motion, yet the particles still show some tendency toward following the first principal curvature direction.

Next in `density.mpg`, the tooth is shown without density controlling rules. As the particles move over time, they tend to accumulate in ridges as a result of following the first principal curvature direction. With the absence of particles in some regions, the shape becomes less clear. With the addition of magnetic repulsion, the distribution of particles becomes much more uniform and the resulting video reveals more shape information.

The next sequence (`size.mpg`) illustrates the effect of changing particle size. When particles are large, they can cover a surface much like splatting, but their motion becomes obscured. When particles are small, they can be difficult to see, and do little to improve perception. The last sequence (`mouse2.mpg`) shows kinetic visualization of the PET data with changing view direction.

## VII. USER STUDY

For a more objective evaluation of the effectiveness of kinetic visualization a user study was conducted. Height field data sets were generated with several randomly placed peaks and valleys, selected examples of which are seen in Figure 16. A static image and kinetic visualization video sequence using the combined rules of PCD following, flocking, and magnetic repulsion were pre-rendered for each data set, with a viewpoint directly above the surface viewing downward. Observers were given the task of identifying the points on each surface they felt were closest and furthest in depth from the viewer. For all data sets, the height and depth of the tallest and shallowest peaks on each surface were at least twice as high or shallow as all others. Subjects were permitted to view each data set for a maximum of 30 seconds, and saw each data set rendered as either a static image with traditional Phong shading, or with kinetic visualization, but never both.

Twenty-two subjects, consisting of both undergraduate and graduate students, were shown fourteen different data sets, half of which were randomly rendered as either a static image or with kinetic visualization. Thus, the combined subject selected 154 minimum and maximum points on surfaces rendered using each method. The results, summarized in Table I, indicate that

Fig. 16. Selected test images from the first user study.

TABLE I
FIRST USER STUDY RESULTS.

| Task | Num. Correct w/ KV | Num. Correct w/o KV |
|---|---|---|
| Find Max | 100 (65%) | 76 (49%) |
| Find Min | 88 (57%) | 67 (42%) |
| Combined | 188 (61%) | 143 (46%) |

subjects were more accurate at selecting both the minimum and maximum points on the surface with kinetic visualization. Of particular interest, we found that kinetic visualization seemed most effective in improving task performance for the more ambiguous data sets with which the subjects were least successful. Although the scope of this user study was fairly limited, we feel the results are extremely promising, particularly since the motion parameters remained constant for all data sets. We have found kinetic visualization to be most effective when the parameters are fine tuned to a particular data set.

In determining the p-value of the combined data (minimum and maximum) we treated the number of correct identifications as the extrema of quantitative data, and then performed a paired sample t-test, matching shape [25]. The null hypothesis stated that there was no difference between using and not using kinetic visualization (the mean difference is zero), versus kinetic visualization being better (the mean is different from zero in the direction of the use of kinetic visualization). The resulting t-test yielded a test statistic of 3.95 with 27 degrees of freedom. The resulting p-value was 0.00025, indicating there was a statistically significant difference in subject performance between using and not using kinetic visualization.

## VIII. Comparison With Oriented Textures

It is worth discussing some of the differences and similarities between our technique and the use of static oriented textures to illustrate shape [6] [26] [27] [23] [28]. Both methods depart from photorealism to provide supplemental cues to aid in the perception of shape. To that end, both techniques make extensive use of principal curvature directions to provide these cues. The two methods require the setting a similar set of parameters; for static textures one needs to specify the size of the texture primitive, the density of those primitives as well as the length it follows a principal curvature direction, which is analogous to the particle size, density and speed parameters used in our work.

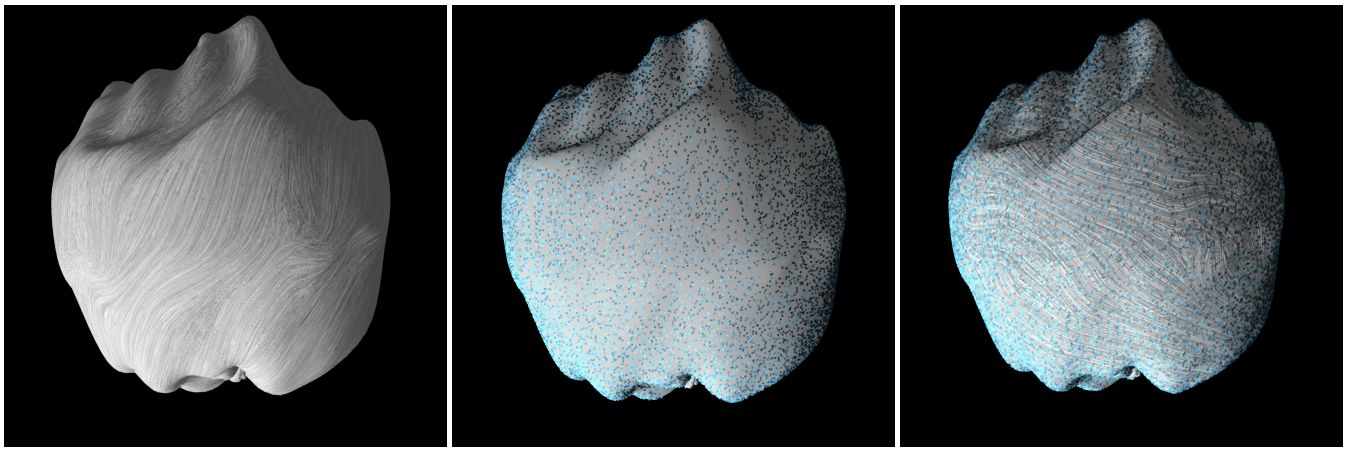
There are, however, significant differences between the two techniques. First, it should be emphasized that our technique differs from texture based methods in its use of the motion of particles to illustrate shape rather than texture. Although we have described rendering optimizations that use texture graphics hardware for improved rendering performance of these particles, our approach is not inherently texture based. A more traditional use of the word 'texture' might be associated with the physical material properties found on the surface of an object, or in computer graphics textures might refer to the application of images to a surface to approximate these material properties. In this respect our method is not based on textures since the particles in our work move over a surface and are independent of any physical texture on the surface itself.

This leads to one advantage of our method over the use of oriented textures. Namely, oriented texture techniques preclude the use of more traditional textures that might be used to illustrate the properties of a surface. For example the use of hatching textures can introduces ambiguity between whether the hatching pattern displayed on a surface is a physical characteristic of the surface (like a weaved basket) or the result of the rendering algorithm itself. Since our method uses motion to illustrate shape, and has a dynamic particles representation, traditional textures can be used in addition to the particles.

Often a priori knowledge of the structure of an object makes supplemental cues for gaining understanding of the overall shape unnecessary. When viewing a tooth data set, the viewer likely has prior understanding of the overall shape of the tooth. Greater ambiguity in shape, however, can exist in smaller substructures that are particular to that individual data set, thus making it desirable to give the user the ability to zoom in on specific features of interest. By using point-based particle primitives that can be rendered with a size defined in screen space, rather than object space, our technique is able to deal with changes in magnification. For example, the user can zoom in to a specific region in the volume and the particle rendering budget will be used only to render particles in that region, while the particles, with their constant screen space size, continue to be effective. For a texture based approach to handle magnification high resolution textures would need to be calculated for smaller scales. A multi-resolution representation similar to that described by Praun et al. [23] for 2D surface models could be applied to deal with such magnifications but could result in impracticably high storage costs from the 3D textures used for volume data.

For future work we would like to conduct further user studies to compare the effectiveness of kinetic visualization with static oriented textures. The video (`orientedtexture.mpg`) shows a surface that has been rendered using oriented textures followed by the same surface rendered using our technique. Snapshots from this animation is shown in Figure 17(a) and Figure 17(b). Since the effectiveness of oriented texture as well as our technique is dependent on a set of rendering parameters, care would need to be taken with respect to the selection of these parameters in order to perform a fair comparison of both techniques when conducting a user study. We suspect that one technique might not always be better than the other and there might be differences depending on the types of tasks a user is asked to perform.

We have begun preliminary research into how our technique can be applied in combination with oriented textures. The video sequence (`texturekinvis.mpg`) shows a surface that has been rendered using oriented textures that follow the second principal direction while kinetic visualization illustrates the first principal direction. A still image of the particles imposed over an oriented texture is shown in Figure 17(c).



(a) (b) (c)

Fig. 17. Our technique shares some similaries with rendering methods that use oriented textures to illustrate shape. Our kinetic visualization technique is fundamentally different in the sense it uses motion rather than texture to illustrate shape. (a) An object rendered using oriented textures. (b) A frame from a kinetic visualization animation (c) A frame from a kinetic visualization animation that has been combined with an oriented texture that illustrates the second principal curvature direction.

## IX. Conclusion

There is a growing interest in making perceptually effective visualizations. In this paper, we summarize our experience with adding visually rich motion cues to the visualizations for increased clarity and expressiveness. While more work is needed, our current results are encouraging, demonstrating that it is feasible and desirable to capitalize on motion cues for the purpose of enhancing perception of 3D shape and spatial relationships.

We have shown that kinetic visualization nicely supplements conventional rendering for illustrating both volumetric and surface data models. We have also shown how the moving particles help reveal surface shape and orientation. By utilizing low-cost commodity hardware, the kinetic visualization system we have built is very affordable. The selective rendering based on particle budget ensures the critical interactivity required for kinetic visualization.

For certain classes of data, however, some limitations in the effectiveness of our technique can be observed. In cases where the principal curvature directions are not well defined, for example flat or spherical regions, the effectiveness of having particles move along a principle curvature direction is limited. The use of optimization strategies, like that described by Hertzmann and Zorin [27] could be used to add direction consistency in these regions, but consideration would also need to be given to avoid smoothing subtle features of interest.

It is clear that our technique is not appropriate for visualizing time varying phenomena. Since the motion of particles in our work is based on the geometric properties of a data set, the motion can give the perception of movement that is contrary to that which is physically occurring. For example, our technique would not be appropriate for visualizing fluid flow since the motion of particles could give a misleading indication of flow direction.

Despite the limitations listed above, we believe that kinetic visualization deserves additional study. Further user studies using a wider variety of tasks and data could provide valuable feedback for improving the technique. We are particularly interested in studying how kinetic visualization affects the performance of real world task using real world data. Conducting user studies comparing kinetic visualization with other visualization techniques, such as oriented textures and rotation, could help to gain understanding of how the different methods affect shape perception, and under what types of conditions the different techniques best suited. This information could be used to create new visualization methods that are even more effective for a wider range of conditions.

Additional future work includes using improved methods for computing principal curvature directions, and accelerating the integrated rendering as much as possible to attain even higher interactivity. It is hoped the power of motion cues will be welcomed by others in helping them to effectively perceive/illustrate complex or ambiguous object shape and spatial relationship.

## Acknowledgements

This work has been sponsored by the National Science Foundation under contract ACI 9983641 (PECASE Award) and through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251. We are grateful to Dr. Juan Jose Vaquero and Dr. Michael Green at the National Institutes of Health, GE Aircraft Engines in Evendale, Ohio, the German Research Council, and SUNY Stony Brook, NY for providing the test data sets. We would like to thank Gabriel Chandler for his assistance in performing the statistical analysis of the results of the user study. We are also grateful to all of the people who participated in our user study.

## References

[1] E. B. Lum, A. Stompel, and K.-L. Ma, "Kinetic visualization: a technique for illustrating 3d shape and structure," in *Proceedings of the Visualization '02 Conference*, 2002.

[2] S. Zeki, *Inner Vision*, Oxford University Press, 1999.

[3] R. A. Andersen and D. C. Bradley, "Perception of three-dimensional structure from motion," *Trends in Cognitive Science*, vol. 2, no. 6, pp. 222–228, June 1998.

[4] S. Treue, M. Husain, and R. A. Andersen, "Human perception of structure from motion," *Vision Research*, vol. 31, pp. 59–75, 1991.

[5] W. T. Reeves, "Particle systems-a technique for modeling a class of fuzzy objects," in *SIGGRAPH '83 Conference Proceedings*, July 1983, pp. 359–376.

[6] V. Interrante, "Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution," in *SIGGRAPH '97 Conference Proceedings*, August 1997, pp. 109–116.

[7] E. B. Lum, K.-L. Ma, and J. Clyne, "Texture hardware assisted rendering of time-varying volume data," in *Proceedings of the Visualization '01 Conference*, 2001.

[8] A. Gooch, B. Gooch, P. Shirley, and E. Cohen, "A non-photorealistic lighting model for automatic technical illustration," in *SIGGRAPH '98 Conference Proceedings*, July 1998, pp. 447–452.

[9] D. Ebert and P Rheingans, "Volume illustration: Non-photorealistic rendering of volume models," in *Proceedings of IEEE Visualization 2000 Conference*, October 2000, pp. 195–202.

[10] D. J. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley, 1996.

[11] H. Wallach and D. N. O'Connell, "The kinetic depth effect," *Journal of Experimental Psychology*, vol. 45, pp. 205–217, 1953.

[12] L. R. Wanger, J. A. Ferwerda, and D. P Greenberg, "Perceiving spatial relationships in computer-generated images," *IEEE Computer Graphics and Applications*, vol. 20, no. 3, pp. 44–58, May 1992.

[13] S. Limoges, C. Ware, and W. Knight, "Displaying correlations using position, motion, point size or point colour," in *Proceedings of Graphics Interface '89*, June 1989, pp. 262–265.

[14] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," in *SIGGRAPH '84 Conference Proceedings*, July 1984, pp. 137–145.

[15] M. Potmesil and I. Chakravarty, "Modeling motion blur in computer-generated images," in *SIGGRAPH '83 Conference Proceedings*, July 1983, pp. 389–399.

[16] W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Motion without movement," in *SIGGRAPH '91 Conference Proceedings*, July 1991, pp. 27–30.

[17] B. Cabral and L. Leedom, "Imaging vector fi elds using line integral convolution," in *SIGGRAPH '93 Conference Proceedings*, August 1993, pp. 263–270.

[18] V. Interrante, H. Fuchs, and S. Pizer, "Conveying the 3D shape of smoothly curving transparent surfaces via texture," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 98–117, April-June 1997.

[19] M. Zwicker, H. Pfi ster, Jeroen van Baar, and M. Gross, "Surface splatting," in *SIGGRAPH 2001 Conference Proceedings*, August 2001, pp. 371–378.

[20] L. Westover, "Interactive volume rendering," in *Chapel Hill Workshop on Volume Visualization*, 1989, pp. 9–16.

[21] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *SIGGRAPH '87 Conference Proceedings*, July 1987, pp. 25–34.

[22] G. Schussman, K.-L. Ma, D. Schissel, and T. Evans, "Visualizing DIII-D tokamak magnetic fi eld lines„" in *Proceedings of IEEE Visualization 2000 Conference*, October 2000, pp. 501–504.

[23] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-time hatching," in *SIGGRAPH '01 Conference Proceedings*, 2001.

[24] A. Van Gelder and U. Hoffman, "Direct volume rendering with shading via three-dimension textures," in *ACM Symposium on Volume Visualizatrion '96 Conference Proceedings*, 1996.

[25] J. M. Utts and R. F. Heckard, *Mind On Statistics*, Duxbury Press, 2002.

[26] A. Girshick and A. Interrante, "Real-time principal direction line drawings of arbitrary 3D surfaces," in *Computer Graphics Visual Proceedings (ACM SIGGRAPH 99 Technical Sketch)*, 1999, p. 271.

[27] A. Hertzmann and D. Zorin, "Illustrating smooth surfaces," in *SIGGRAPH 2000 Conference Proceedings*, August 2000, pp. 517–526.

[28] S. Kim, H. Hagh-Shenas, and V. Interrante, "Showing shape with texture: two directions are better than one," in *Proceedings of Human Vision and Electronic Imaging VIII (SPIE)*, January 2003.