Rendering Particles for 5D Scalar Fields in a Virtual Environment: The Need for Speed

Rhonda J. Vickery*
Mississippi State University
rvickery@erc.msstate.edu

Timothy R. Keen**
Naval Research Laboratory
keen@nrlssc.navy.mil

Robert J. Moorhead*
Mississippi State University
rjm@erc.msstate.edu

Joerg Meyer†
University of California, Irvine
jmeyer@uci.edu

Randy J. Brou*
Mississippi State University
rjb2@erc.msstate.edu

Ashley M. Noble*
Mississippi State University
amr@erc.msstate.edu

Joel P. Martin*
Mississippi State University
jmartin@erc.msstate.edu

Stephanie M. Doane*
Mississippi State University
sdoane@erc.msstate.edu

Keywords: Particle Rendering, Texture Synthesis, Texture Generation, Computer Graphics, Scientific Visualization

Abstract

This work presents a Synthesized Cell Texture (SCT) volume visualization algorithm for multiple scalar value fields, where a specified maximum slice resolution is used to scale the multi-scalar field values for each 3D cell to the maximum values found throughout the data set. The values are randomly distributed as particles varying in number, size, color, and opacity within a 2D synthesized texture. This approach facilitates viewing of closely spaced layers commonly found in sigma coordinate grids. Computation time and texture memory are traded off against the number of geometric primitives which must be sent through the graphics pipeline of the host system. The system is optimized for deployment in a four wall CAVE-like virtual environment.

The SCT method is compared with two common basic particle representations: flat-shaded colormapped OpenGL points and quadrilaterals. Performance statistics show the SCT method to be up to 44 times faster, depending on the volume to be displayed and the host system. The SCT method has been successfully applied to oceanographic sedimentation data, and can be applied to multiple scalar fields in other problem domains as well. Future enhancements include the extension to time-varying data and parallelization of the texture synthesis component to reduce startup time.

*ERC, PO Box 9627, Mississippi State, MS 39762
**Oceanography Division, Stennis Space Center, MS 39529
†Department of Electrical Engineering and Computer Science, 644E Engineering Tower, Irvine, CA 92697-2625

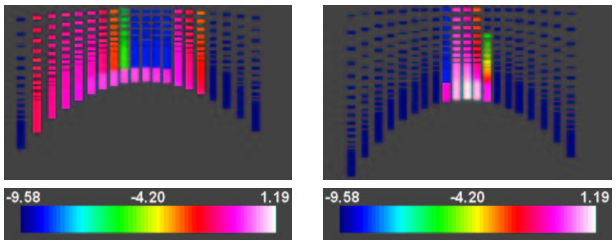
INTRODUCTION

Modeling and simulation of complex processes often produce scalar, or single valued, data. Rather than analyzing the raw data values on a point-by-point basis, researchers have turned to ways of visually representing entire regions of data, such that relationships between variables can be examined. Usually only one scalar field is present, but there are physical problem domains where it is desirable to visualize several related scalars at once. These are often referred to as "profiles" with many different scalar values represented at one grid point (up to 20 for some sedimentation data), and whose values are of interest both individually and as part of the whole profile. This extra dimension is problematic for off-the-shelf visualization software, and has until recently only been visualized as a single combined entity throughout a volume, or as individual scalar values. The full profiles themselves have only been visualized at individual points or columns of points aligned in the z direction.

For many of these applications, the scalar field represents a quantity of suspended material that is expressed as mass per unit volume (sedimentation) or mass of suspended particulate per mass of suspending volume (for many atmospheric processes). Although this puts all quantities on an equal basis for comparison, it does not give a direct indication of how many particles of different sizes are actually suspended. What this means is that for each physical point in space, or grid point in the model, there are concentration values for a number of different grain sizes (the multi-scalar aspect). The

combined effect is important for the production of accurate model results, but the added dimension creates difficulties for existing visualization methods. In addition to longitude, latitude, depth (or altitude), and time, there is now a fifth dimension, grain size, making it difficult to display values individually using conventional methods.

As an example, point profiles of the overall suspended sediment concentration (SSC) have been used for visualization of sedimentation processes [1, 2]. In Figure 1a, the sediment concentration values are colormapped on a log_{10} scale for a column of water at a grid location on a raised area within a shallow water region (or hill location). The columns of points indicate the individual SSC for each grain size from smallest (left) to largest (right), where zero values are dark blue (black in greyscale). In this case the largest grain size represented is 3000 times larger than the smallest, and so concentration values very close to zero for the smaller grain sizes are still significant. Figure 1b shows a similar profile for a deeper grid location within the same shallow water region (hole location). To the novice analyst, it is not clear how much sediment, in terms of relative numbers of suspended particles, is represented from these views, and this method does not display an integrated view for all grid locations throughout a volume. Given the densities and particle volume information of the suspended particulates, the concentration values can be converted to numbers of particles per unit volume. These quantities can then be rendered using multidimensional multivariate (mdmv) methods [3], including those for direct volume or particle rendering. For sedimentation, colormapped 2D (horizontal) z-layers are another method commonly used to show combined or single valued SSC.



(a) (b)

Figure 1: (a) point detail view with a log_{10} colormap legend showing SSC at a grid location with many smaller particles [2], and (b) Point detail view with a log_{10} colormap legend showing SSC at a grid location with fewer larger particles [2].

The problem domains just described are good candidates for exploration in our four-wall CAVE-like virtual environment (VE) [4], especially for combination views involving concentrations, vector quantities, and other scalar fields [1]. However, rendering concentrations as particles at interactive frame rates using traditional methods (such as texture-mapped primitives or basic points [5, 6]) is computationally expensive and can easily overload the VE graphics pipeline. Therefore, this work describes an alternate means of visualizing multi-scalar data with the introduction of the Synthesized Cell Texture (SCT) algorithm. This hybrid technique indicates the added dimension of a scalar profile variable within a three-dimensional (3D) cell in terms of relative quantities of different size particles. The SCT algorithm can display the additional information throughout a prescribed volume at a desired slice resolution (given existing hardware constraints), and is compared with two basic particle rendering methods: flat-shaded colormapped points and quadrilaterals. Both methods can be considered simpler (and faster) forms of the general texture-mapped primitives approach. Comparison with other mdmv methods requires additional considerations for perceived effectiveness with human subjects, and will be presented in a future publication.

The remainder of this paper is organized into sections on previous work, implementation details for the SCT algorithm, points, and quadrilaterals, and a comparative analysis of the results. Enhancements to the SCT algorithm are presented, including optimizations to reduce startup time and incorporate time-varying data. Finally, conclusions are drawn and future directions for research are described.

PREVIOUS WORK

Effectively representing mdmv scalar fields is an active area of research in information visualization. Two important goals are: (*i*) to discover the values for data within a region, and (*ii*) locate data with specific values [7]. Good visual methods often help the researcher find areas of interest and correlations between variables. With the ability of computers to convey more information at once comes a challenge to determine what visual attributes are best perceived by humans [8]. Hence evaluating the effectiveness of a technique is dependent on perception-based user studies [9], as well as rendering speed and memory efficiency.

An excellent overview and classification scheme for mdmv visualizations is presented by Nielson et al. [3], although many others are available. Statistical and information analysis motivates the use of various 2D and 3D graphical methods, with the objective of conveying information about m-dimensional dependent variables (one convention terms this the multivariate aspect) for n-dimensional independent variables (the multidimen-

sional aspect). For our sedimentation application, SSC represents the dependent variable for the 5D independent variables of longitude, latitude, depth, time, and grain size. Animation is often used to convey changes in dependent variables over time, but displaying SSC values for the remaining four dimensions in 3D space is more problematic. One category of mdmv methods encodes data information to color and geometric attributes of an object (often termed a glyph or icon) [10].

In terms of rendering particle information, a quadrilateral can be considered a glyph with the attributes of color (four components), shape, size, and texture. Any number of these attributes could be mapped to the magnitude of a different scalar field, but perception-based studies determine which attributes can be effectively utilized [8]. Flow visualization often uses texture-mapped quadrilaterals (or basic colormapped points) to represent particles [5, 6, 11–13]. The SCT algorithm incorporates the concept of a glyph at two levels. In this study, an SCT glyph for a particle is considered to be a $4x4$ array of pixels from a 2D texture with the same attributes as a quadrilateral. However, an SCT glyph could also represent the information for a 3D cell volume (containing many particles) at a spatial location. Future studies will compare the SCT algorithm using the cell-based glyph with other mdmv methods appropriate for that representation [14, 15].

Particles for flow visualization may also be rendered by some number of pixels in an overall texture that are then advected over time. These include such methods as spot noise, line integral convolution (LIC), and texture advection [16–20]. These pixel-based particles are generally massless and are injected into the flow at a specific time. Any size, color, or shape variability is only used to indicate direction and orientation of flow and is not tied to any particular scalar field.

Scalar fields are also commonly represented by volume visualization methods. Volume visualization generally involves the rendering of volumetric data sets representing 1D scalar quantities at specific points [21, 22]. The challenge is to classify what the scalar values represent (such as tissue or bone) as indicated by different colors and opacities. Several methods approximate the underlying integral defining the blending of values (such as splatting or texture-based methods), or show 2D slices of the volume (such as projection-based methods), or 3D surfaces [22]. Research has also been done on trying to show two or three independent variables at once in a volume visualization [23]. More work is needed on effectively representing larger numbers of scalars simultaneously. The SCT algorithm can be considered a volume visualization method, but with the emphasis on generation of the colors and opacities of a multi-scalar field, rather than on the classification of a single scalar field.

The SCT algorithm is unique since it can show a number of different size particles in an integrated, overall manner over a chosen volume of cells. By displaying particles at the highest resolution possible (i.e., using one or few pixels per particle with varying opacity), a greater ratio of the number of actual particles present to the number that can be displayed can be realized. Where there are large differences in numbers of particles, this texture-based method can show values in a greater range than two other commonly used glyph-based methods, colormapped quadrilaterals and points. The SCT algorithm allows more detail information to be shown because of the larger pool of particles that can be displayed at one time. The maximum number of glyphs is utilized, given the limitations of the hardware and desired slice resolution chosen. Comparisons with these other glyph-based methods are described in this work and show the effectiveness of the SCT volume visualization technique. Data from coastal zone sedimentation modeling serves as a testbed for the applicability of this algorithm.

Although the main variable of interest in the sedimentation domain is concentration, the SCT method can be applied to any scalar where different characteristics can be represented by four parameters: glyph color, size, and opacity, as well as number of glyphs per category (bin).

IMPLEMENTATION

In this discussion, the term **cell** refers to a 3D volume immediately surrounding a grid location in the physical domain with dimensions $\Delta x \Delta y \Delta z$. A **slice** refers to the 2D texture-mapped primitive centered at the grid location as described by Vickery in [2] and shown in Figure 2. The 2D texture is generated from information about the physical scalar quantities within a 3D cell. The term **pixel** refers to the smallest unit of the 2D texture that has both color and opacity. Since the grid points are packed so densely in the z direction close to the sea floor (our main area of interest), we can visualize the physical scalar quantities for an entire 3D volume as layers comprised of 2D slices. The term **glyph** will refer to the entity that represents a scalar value for a particle, whether it is a $4x4$ grouping of pixels in the texture, a colormapped point primitive specified in the computer graphics language known as OpenGL (also known as an OpenGL point), or a colormapped quadrilateral.

The first stage of the SCT method provides a means of creating a visualization of a volumetric multi-scalar field that can be specified in terms of a maximum slice resolution (or density of represented values). This maximum resolution is used to scale the multi-scalar field

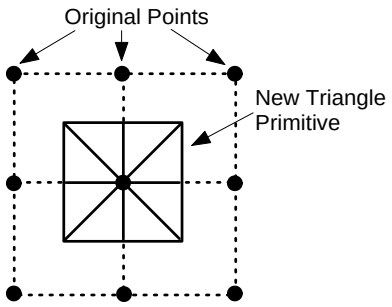


Figure 2: New grid cell with eight triangles (trifan), superimposed on the original grid [2].

for the data over the entire volume, including all time series. Therefore, a prescribed volume of data for any time step can be displayed and compared relative to the same scale with other volumes within the same time step or across multiple time steps. The values within a 3D cell are positioned as non-overlapping particle glyphs equally distributed within a 2D region of size $\Delta x \Delta y$ centered at the original grid location. In other words, this algorithm scales the data for each cell to the maximum values found throughout the data set, and then randomly distributes the values within a flattened 2D region. For this reason, it will be referred to as the Scaling and Distribution or SAD algorithm. This approach facilitates viewing of closely spaced layers as commonly found in sigma coordinate grids (in this case, logarithmic spacing in the vertical direction with 31 levels). This portion of the algorithm can be applied regardless of how the multi-scalar entities are rendered (i.e. as parts of a 2D texture-mapped slice, colormapped OpenGL points, or colormapped quadrilaterals).

The second stage provides a hardware texture mapping algorithm to render the multi-scalar values. In this approach, a 2D texture for each slice is synthesized from the location information from the first step, and then rendered at the correct grid location within the volume. This method trades off computation time (to synthesize the texture) and texture memory against the number of geometric primitives which must be sent through the graphics pipeline of the host system. Hence the name Synthesized Cell Texture or SCT algorithm is appropriate. The SCT algorithm incorporates the SAD algorithm as part of the rendering of the volume visualization. Therefore, SCT will also refer to the volume visualization algorithm in general.

The first version of the SCT algorithm converted the twenty scalar values at each grid point represented in units of Kg/m^3 to numbers of particles of each bin[1] [2]. A 2D texture mapping scheme was employed to display the particles where a texture for each cell layer was synthesized to show the relative numbers of particles as *same size pixels* using a different color for each bin (twenty colors in all). This method suffered from two major shortcomings: particles of vastly different sizes were displayed as the same size, and there were too many colors to distinctly distinguish between particle bins.

This original algorithm was recently enhanced to allow a custom bin consolidation scheme to be applied to reduce the number of bins to sixteen (or less), such that a glyph with a $4x4$ footprint could be used to show grain size differences[2] [24, 25]. Bins may also be grouped into more general categories to reduce the number of colors that must be represented, or an automatic color assignment algorithm can be used. Four colors represented the sediment types: clay, silt, sand, and gravel [25]. A particle sizing scheme was also implemented such that particles from different bins are represented as varying size particles. This was accomplished by using a $4x4$ pixel glyph in the SCT algorithm as the basic particle footprint and coloring from one to sixteen pixels to show the relative sizing of particles.

Figure 3 shows the results of these enhancements for a single texture slice for a cell at the hill location. Here bin differences are indicated by particle size and color. Large numbers of clay particles are shown by one pixel green glyphs, as well as several sizes of silt particles (red), and very few larger sand particles (cyan). A breakdown of the bin categories can be seen in Table 1. A twelve layer volume visualization using the enhanced algorithm with 3D point detail profiles for both hill and hole locations is illustrated in Figure 4 and Figure 5. Careful observation of the volume visualization shows the difference between a grid location where many smaller particles result in very low sediment concentration (hill), and one where fewer larger particles make up a much larger sediment concentration (hole)[3]. This is contrary to what one might think from just looking at the detail profiles (see Figure 1). This is due to the differences in the grain sizes (diameters of an assumed spherical particle) between bins, and the part it plays in the mass per volume computation for sediment concentration (Kg/m^3) [2, 25].

In the latest version of the SCT method, the SAD and SCT stages are modularized and separated to allow comparisons with two flat-shaded colormapped

[1] In the sedimentation data, each bin represents values of a particular grain size (diameter) of sediment.

[2] For the sedimentation data, this is the USDA soil texture classification that reduced the bins to fourteen, although a generic bin grouping or reduction algorithm could also be used.

[3] The color version of this paper illustrates this much better and is available online at http://www.erc.msstate.edu/∼rvickery/publications.html

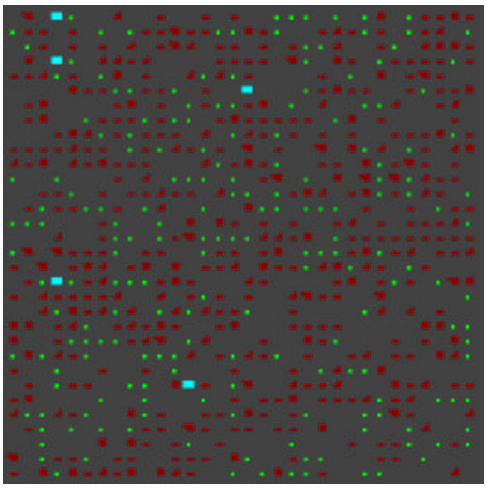


Figure 3: Difference in color separation and particle size is shown in a flat single bottom layer texture synthesized by the enhanced algorithm for the hill location [25].

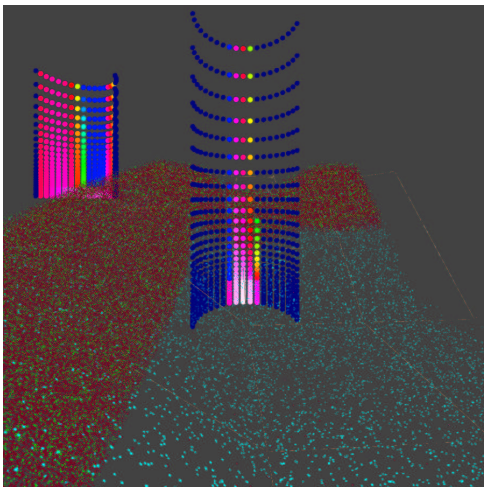


Figure 4: Twelve layer volume visualization using the enhanced algorithm with point detail profiles for both hill and hole locations [25].

Table 1: Bin Categories with pixel and color assignments.

Bins / Num Pixels	Soil Type	Color
1	Clay	Green
2-5	Silt	Red
6-10	Sand	Cyan
11-16	Gravel	Dark blue

representations for particles: OpenGL points (Points) and quadrilaterals (Quads). Both methods can be considered simpler (and faster) forms of the general texture-mapped primitives approach. Detailed results show where the performance of the methods differ. The system is optimized for deployment in a four wall CAVE-like virtual environment (VE). Performance data are obtained using extensive built-in graphics statistics gathering features provided by the OpenGL Performer^{TM} toolkit developed by Silicon Graphics, Inc. (SGI). This scene graph based software toolkit is designed to help optimize applications and discover bottlenecks running on SGI supported platforms [26, 27]. Initial computation times for each of the three methods are recorded using the SGI high-resolution *syssgi* hardware clock with sub-microsecond accuracy and accessible through OpenGL Performer^{TM}[4]. Although the Points representation is included in the performance comparisons, it is not suitable for actual implementation in our application since the points are not all the same approximate distance from the viewer. OpenGL points are rendered to a specific "point size" which always take up the same number of pixels on the display regardless of the proximity of the viewer (making them change in size relative to surrounding geometry depending on the viewer location). OpenGL points may be applicable in other situations where the particles are all viewed from approximately the same distance.

The test configurations were chosen to investigate the limitations of the three algorithms in the following scenarios:

1. A large $\Delta x \Delta y$ shallow water region that maximizes the amount of texture memory used that includes both dense and sparse regions of particles, but fewer layers (which means less blending of layers during rendering). Methods which use primitives for individual particles would do less work in sparse regions. Machine A was chosen for this purpose since it has the largest amount of texture memory and the most advanced SGI graphics pipeline (see Table 2). This region contains 252 grid locations per z-direction layer and tests were run from 1 to 8 layers.

2. A smaller shallow water region containing dense concentrations of particles over the full 31 layers of the dataset (132 grid locations per z-direction layer). This configuration maximizes the number of glyphs and the blending of layers during rendering. Since the application is designed for our CAVE-like VE, Machine B is the testbed for this

[4]This is much more precise than the commonly used time-of-day clock which only has an accuracy of 1-10 milliseconds, depending on the hardware platform.

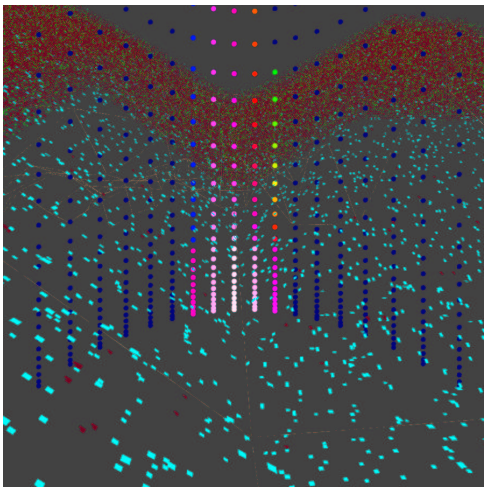


Figure 5: Closeup of the hole location [25]. Note that the smaller grain size particles (shown higher up on the hill) are stripped away at the deeper locations.

scenario. Data was collected for layers 1 through 8, then every fourth layer through 31.

3. The same smaller shallow water region on Machine B, but studying the effects of increasing slice resolution on rendering for 3 layers. Data was collected for resolutions of $16x16$ through $128x128$.

Each test case was run 5 times and the times were averaged. Since OpenGL has the requirement that the length and width of texture images must be a power of 2, the slice resolutions were chosen accordingly [28].

Table 2: Machine configurations for benchmark tests.

	Machine A	Machine B
Type	Onyx 2	Onyx 2
Graphics	Infinite Reality 3	Infinite Reality
Num Processors	4	8
RAM (GB)	4	4
Texure Memory (MB)	512	128
Usage	High-end Desktop	CAVE-like VE

RESULTS

The frame rate comparison in Figure 6 shows that the SCT algorithm executes at 66 fps for up to 5 layers of the shallow water region, then decreases to 33 fps for layers 6 and 7. This occurs because OpenGL Performer^{TM} attempts to match the frame rate of the scene rendered to the video refresh rate [26]. For Machine A, this rate is 66 Hz. Once the frame rate drops below 12 fps, the rate is not affected by the video refresh rate, and is based simply on the number of times the scene can be rendered during a second.

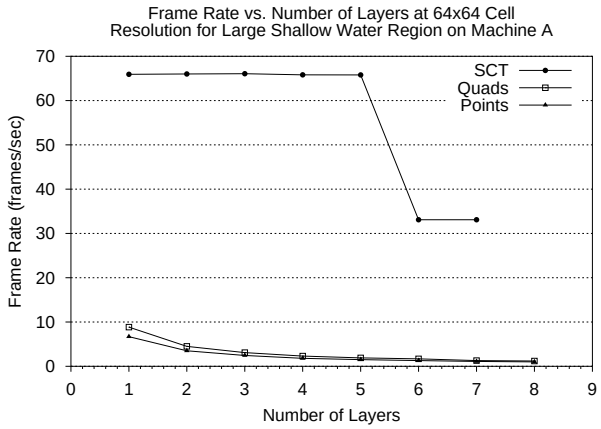


Figure 6: Frame Rate vs. Number of Layers for Machine A.

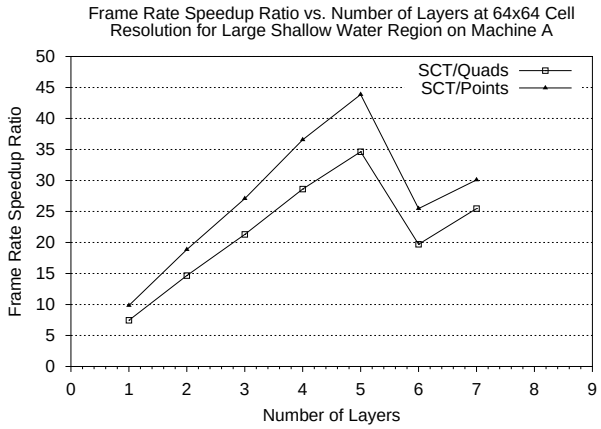


Figure 7: Frame Rate Speedup Ratio vs. Number of Layers for Machine A.

As shown in Figure 6, the Quads algorithm rendered the first layer of the configuration at just under 9 fps, and the Points algorithm at around 7 fps. Note that the Quads method is generally faster than Points since the Infinite Reality graphics pipeline is optimized for triangles (a quadrilateral is internally split into 2 triangles), and the pipeline must do extra work to draw each point to the point size specified in the Points algorithm. In contrast to the SCT algorithm, neither the Quads nor Points methods could produce acceptable frame rates for two or more layers.

The results can also be shown in terms of a frame rate speedup ratio of SCT vs. Quads or Points, which can be computed in terms of fps_{SCT}/fps_{Other}. In Figure 7, the frame rate speedup ratio for SCT vs. Quads varies from 7 to just under 35, with higher speedups occurring when the Quads method bogs down the graphics

pipeline when trying to render more layers. For SCT vs. Points, the frame rate speedup ratio varies from 10 to 44.

Another perspective can be seen in Figure 8 and Figure 9. This series of runs was performed in the VE configuration on Machine B for the shallow water region. Figure 8 shows frame rate vs. number of layers for the SCT, Quads, and Points algorithms for a low slice resolution of $16x16$. In this case, the frame rate for all methods is considerably affected by the amount of blending required for the large number of glyphs displayed in a vertical column. The Quads and Points algorithms can maintain a frame rate above 10 fps for up to 5 layers. At 8 layers, a frame rate of around 6 fps is still navigable in the VE, but difficult. The SCT algorithm starts as high as 48 fps for 1 layer, stays above 10 fps for up to 10 layers, and 6 fps for up to 16 layers. At a slice resolution of $16x16$, each layer adds about 2 MB of texture memory, and Machine B has enough to view all 31 layers. The end result is that the SCT algorithm is still up to 2 times faster than Quads or Points in cases where the texture memory is not filled to capacity, but where there may be many glyphs to display (see Figure 9).

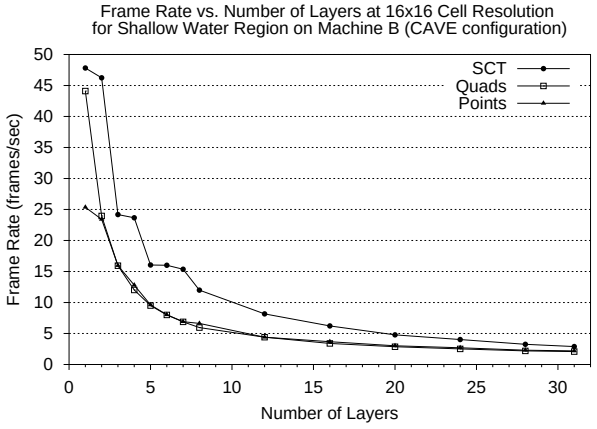


Figure 8: Frame Rate vs. Number of Layers for Machine B.

Figure 10 shows the same VE shallow water region setup for 3 layers at differing slice resolutions of $16x16$ through $128x128$. The Quads and Points algorithms can maintain a frame rate of 16 fps at $16x16$, but only 4 to 5 fps at $32x32$. In contrast, the SCT algorithm can maintain a frame rate of 24 fps up to a resolution of $64x64$. In other words, the SCT algorithm can display 4 to 16 times the number of particles at a faster frame rate than the other two algorithms (for this particular case). In cases where there are very few particles in a particular bin, the SCT algorithm is more likely to

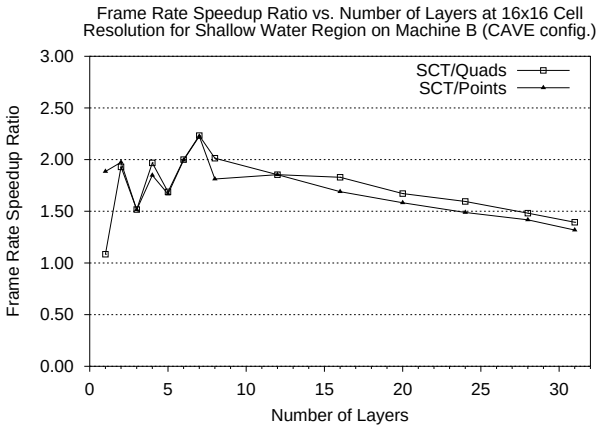


Figure 9: Frame Rate Speedup vs. Number of Layers for Machine B.

be able to represent those particles with a higher resolution than is possible with the other algorithms. The numbers of glyphs computed for each bin at each resolution for this configuration are shown in Table 3[5]. Note that bin 7 is not represented at the $16x16$ resolution, but is represented at $32x32$ and higher. This indicates that with a higher slice resolution, the SCT algorithm can display values in a greater range than the Quads or Points methods at an acceptable frame rate. This capability allows more detail information to be shown because of the larger pool of particles that can be displayed at one time, and improves with increasing slice resolution.

Table 3: Number of actual consolidated glyphs used in volume by resolution and particles per glyph (PPG) for three layers of the shallow water region. Values are shown for the first seven bins.

Bin	16x16	32x32	64x64	128x128
1	12302	49212	196885	787618
2	18088	72463	289876	1.15953E+06
3	5100	20345	81392	325642
4	4713	18840	75332	301350
5	1663	6522	26171	104574
6	404	2110	8771	35073
7	0	73	524	2342
Total	42270	169565	678951	2.71613E+06
PPG	6.18182E+09	1.54104E+09	3.84867E+08	9.62051E+07

It is useful to note that frame rate comparisons do not tell the whole story. Although the SCT algorithm can achieve a higher frame rate than either Points or Quads, it does so at the expense of more memory and startup processing time to synthesize the texture. Fig-

[5]For this particular dataset, the consolidation scheme resulted in values for only seven of the bins. Other datasets, such as those for hurricane weather events, would have values in all of the bins, representing more of the larger size particles.

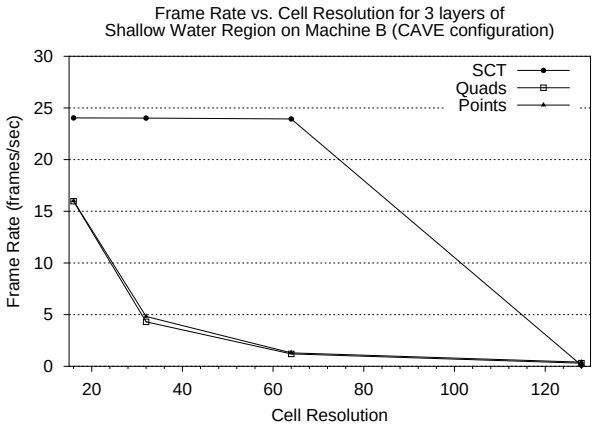


Figure 10: Frame Rate vs. Slice Resolution for Machine B.

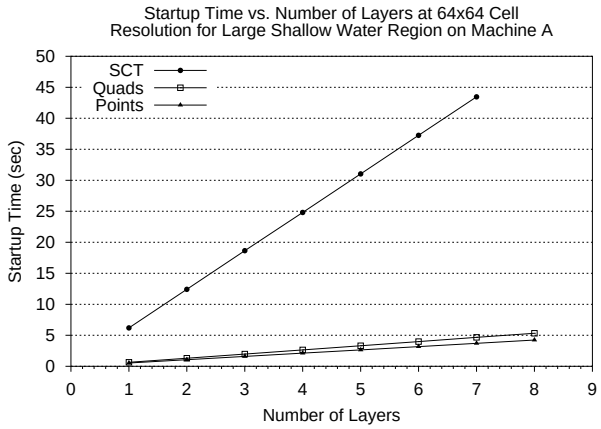


Figure 11: Startup Time vs. Number of Layers for Machine A.

ure 11 shows the startup times for execution of all three algorithms. The time required per layer varies linearly with all three methods, although the slope for the SCT algorithm is much steeper (6.20 sec/layer for SCT, 0.67 sec/layer for Quads, and 0.53 sec/layer for Points). This slope can be converted to a general slope of seconds required per megabyte of memory: SCT requires 0.10 sec/MB, Quads requires 0.06 sec/MB, and Points requires 0.18 sec/MB (see Figure 12). This may seem counterintuitive, but in reality all pixels in the SCT texture must be assigned values, even if they do not represent actual glyphs for particles (in which case they would be transparent). The Quads and Points algorithms only use memory for the actual glyphs rendered. The fact that the Quads algorithm takes less time per megabyte than the Points is due to the similar amount of time it takes for both methods to locate the glyphs, with only slightly more work required to place four closely spaced points for a quadrilateral than a single point. Therefore, the Quads algorithm requires more memory, but the time per MB is less. Figure 13 confirms the additional time required for the SCT algorithm: it is about 9.4 times slower than the Quads, and 11.7 times slower than the Points.

One goal of this study was to determine the limitations of the SCT algorithm, and Machine A was used for this purpose. There was only one case where the SCT algorithm did not render the visualization, and this occurred when it could not complete the 8 layer configuration because a process maximum shared memory limit of 500 MB was exceeded (see Figure 6). This was due to an OpenGL Performer^{TM} memory mapping problem during the attempted allocation of about 500 MB of texture memory, as indicated in Figure 12 and documented in the release notes [29].

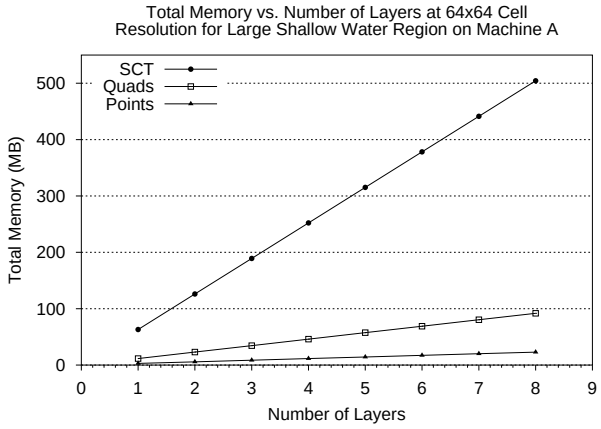


Figure 12: Total Memory vs. Number of Layers for Machine A.

The SCT algorithm has been shown to take considerably more memory and startup time than either of the other two methods. However, both aspects can be easily justified as reasonable for the increased frame rate during scene rendering. Texture memory is available in ever increasing amounts on new computers expressly for this purpose [30], and massive amounts of main memory are utilized to reduce file access latencies. At the point where the texture is computed, there is only color and transparency assigned based on bin type for each glyph. This portion could easily be split by cell and partitioned among several processors as described in the next section on future enhancements.

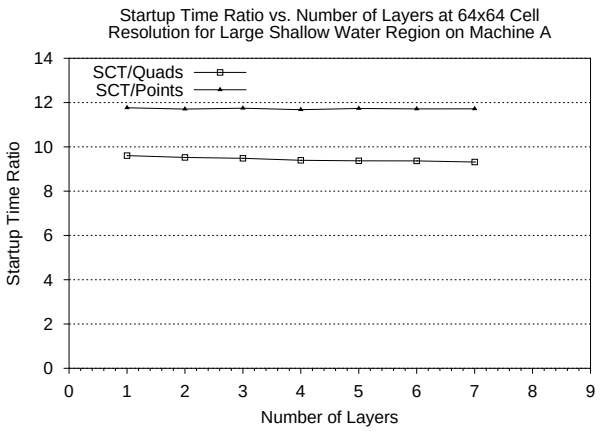


Figure 13: Startup Time Speedup Ratio vs. Number of Layers for Machine A.

FUTURE ENHANCEMENTS

Three straightforward enhancements to the SCT algorithm include adding a variable size glyph, parallelizing the texture synthesis portion of the algorithm, and extending the method for time-varying data. Implementing a variable size glyph (such as $3x3$ or $4x5$) is complicated by the OpenGL optimization requirement that the length and width of a texture image be a power of 2 [28]. The leftover pixels computed during the texture synthesis stage would need to be randomly distributed and made transparent. This would allow better utilization of the pixels in the texture and more bins to be represented.

The parallelization of the texture synthesis portion of the SCT algorithm can be accomplished by partitioning the total cells to be textured among available processors on the host and neighboring computers. For instance, the longest startup time for the SCT algorithm was just under 45 seconds for 1764 cells of the shallow water region. If the task was simply split among 20 processors, then each one would compute the texture for about 88 cells. Even with the added overhead of communication and other process loads, it seems reasonable that the overall startup time of the SCT algorithm could be reduced to less than 5 seconds.

Extending the SCT algorithm for time-varying data requires that the bin information for each glyph be saved along with the texture for the current time step, in order to compute the differences in the texture for the next time step. The same texture memory can be used and only glyph locations within the texture that must change are affected.

CONCLUSIONS

In this study the SCT method was compared with two glyph-based representations: OpenGL points and quadrilaterals. Preliminary performance statistics gathered in the comparisons show the SCT method to have an increase in rendering speed of up to 44 times faster than the other methods, depending on the volume to be displayed and the host system. For a given frame rate, performance data show that the SCT algorithm can display from 4 to 16 times the amount of information of the Quads or Points algorithms at a faster frame rate in the VE.

The SCT method has been successfully applied to oceanographic sedimentation data with up to twenty scalar values per grid point. In this case the grid is regular in the xy direction and sigma coordinate in the z direction (and exponentially distributed with closely packed layers close to the bottom of the grid). The method may be applied to other problem domains as well. Future publications will describe its use in the visualization of dust aerosol transport in global circulation models [31].

Future work includes adding the enhancements described, and performing a user study on the effectiveness of the SCT representation as an mdmv method.

References

[1] Keen, T. R.; R. J. Vickery; P. Flynn; R. H. Stavn; W. McBride, 2001. "Scientific Visualization of Sediment Dynamics in the Bottom Boundary Layer." In *7th Int. Conf. Estuarine and Coastal Model., Proc.* St. Pete Beach, Florida.

[2] Vickery, R. J.; T. R. Keen; R. J. Moorhead; R. J. Brou; D. W. Carruth; S. M. Doane, 2002. "Volume Visualization of 5D Sedimentation Models." In *Visualization and Data Analysis 2002*, edited by R. F. Erbacher; P. C. Chen; M. Groehn; J. C. Roberts; C. M. Wittenbrink. San Jose, CA, volume 4665 of *Proceedings of SPIE*, 165–176.

[3] Nielson, G. M.; H. Hagen; H. Müller, editors, 1997. *Scientific Visualization: Overviews, Methodologies, and Techniques*. IEEE Computer Society, Los Alamitos, CA.

[4] Cruz-Neira, C.; D. J. Sandin; T. A. DeFanti, 1993. "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE." In *SIGGRAPH 93 Conference Proceedings*, edited by J. T. Kajiya. ACM SIGGRAPH, Addison Wesley, Annual Conference Series, 135–142. ISBN 0-89791-601-8.

[5] Lum, E. B.; K.-L. Ma, 2002. "Interactivity is the Key to Expressive Visualization." *ACM SIGGRAPH Computer Graphics 36*, no. 3, 5–9.

[6] Kuester, F.; R. Bruckschen; B. Hamann; K. I. Joy, 2001. "Visualization of Particle Traces in Virtual Environments." In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. ACM Press, Baniff, Alberta, Canada, 151–157.

[7] Wegenkittl, R.; H. Löffelmann; E. Gröller, 1997. "Visualizing the Behavior of Higher Dimensional Dynamical Systems." In *Proceedings Visualization '97*. ACM Press, New York, NY, 119–125.

[8] Ware, C., 2000. *Information Visualization: Perception for Design*. Morgan Kaufmann, San Francisco.

[9] Ferwerda, J. A., 2002. "Psychometrics 101: How to Design, Conduct, and Analyze Perceptual Experiments in Computer Graphics." ACM SIGGRAPH 2002 Course.

[10] Levkowitz, H., 1991. "Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Parameters." In *Proceedings Visualization '91*. IEEE Computer Society Press and ACM, 164–170.

[11] Max, N.; R. Crawfis; B. Becker, 1995. "Applications of Texture Mapping to Volume and Flow Visualization." In *Graphicon 95*.

[12] Max, N.; B. Becker, 1996. "Flow Visualization Using Moving Textures." In *Proceedings of ICASE/LaRC Symposium on Visualizing Time Varying Data*, edited by D. Banks; T. Crockett; S. Katy. NASA Conference Publication, volume 3321, 77–87.

[13] Schroeder, W.; K. Martin; W. Lorensen, 1998. *The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics*. Prentice-Hall PTR, Upper Saddle River, NJ 07458, 2nd edition. ISBN 0-13-954694-4.

[14] Taylor, R., 2002. "Visualizing Multiple Fields on the Same Surface." *IEEE Computer Graphics and Applications 22*, no. 3, 6–9.

[15] Laidlaw, D. H., 2001. "Loose, Artistic "Textures" for Visualization." *IEEE Computer Graphics and Applications 21*, no. 2, 6–9.

[16] de Leeuw, W.; R. van Liere, 1998. "Comparing LIC and Spot Noise." In *Proceedings Visualization '98*. IEEE Computer Society Press. ISBN 1-58113-106-2, 359–365.

[17] Heidrich, W.; R. Westermann; H.-P. Seidel; T. Ertl, 1999. "Applications of Pixel Textures in Visualization and Realistic Image Synthesis." In *ACM Symposium on Interactive 3D Graphics*. ACM, 127–134.

[18] Jobard, B.; W. Lefer, 1997. "The Motion Map: Efficient Computation of Steady Flow Animations." In *Proceedings Visualization '97*. IEEE Computer Society Press and ACM, 323–328.

[19] Jobard, B.; G. Erlebacher; M. Hussaini, 2000. "Hardware-Accelerated Texture Advection for Unsteady Flow Visualization." In *Proceedings Visualization '00*. IEEE Computer Society Press and ACM, 155–161.

[20] Rezk-Salama, C.; P. Hastreiter; C. Teitzel; T. Ertl, 1999. "Interactive Exploration of Volume Line Integral Convolution Based on 3D Texture Mapping." In *Proceedings Visualization '99*. IEEE Computer Society Press and ACM, 233–240.

[21] Kaufman, A., editor, 1991. *Volume Visualization*. IEEE Computer Society Press, Los Alamitos, CA.

[22] Meißner, M.; J. Huang; D. Bartz; K. Mueller; R. Crawfis, 2000. "A Practical Evaluation of Popular Volume Rendering Algorithms." In *Proceedings Volume Visualization and Graphics Symposium 2000*, edited by S. N. Spencer. ACM SIGGRAPH, New York, NY, 81–90. ISBN 1-58113-308-1.

[23] Crawfis, R. A., 1995. "New Techniques of the Scientific Visualization of Three-Dimensional Multi-variate and Vector Fields." Ph.D. thesis, Lawrence Livermore National Laboratory, University of California, Livermore, CA.

[24] Pfannkuch, H. O.; R. Paulson. *Grain Size Distribution and Hydraulic Properties*. http://www.cs.pdx.edu/ ian/geology2.5.html.

[25] Vickery, R. J.; T. R. Keen; R. J. Moorhead; J. Meyer; R. J. Brou; A. M. Noble; J. P. Martin; S. M. Doane, 2002. "Interactive Poster: Effects of Relative Particle Sizing and Bin Consolidation Enhancements to a 5D Volume Visualization Algorithm." In *Visualization 2002 DVD*. IEEE Computer Society Press and ACM, Boston, MA.

[26] Eckel, G. *IRIS Performer Programmer's Guide*. Silicon Graphics, Inc. Document Number 007-1680-040.

[27] Rohlf, J.; J. Helman, 1994. "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics." In *SIGGRAPH 94 Conference Proceedings*, edited by A. S. Glassner. ACM SIGGRAPH, Addison-Wesley, Annual Conference Series, 381–394. ISBN 0-89791-667-0.

[28] Woo, M.; J. L. Neider; T. R. Davis; D. R. Shreiner, 1999. *OpenGL Programming Guide, Third Edition*. Addison-Wesley, Reading, Mass.

[29] *OpenGL Performer Version 2.5 Release Notes*.

[30] Rezk-Salama, C.; K. Engel; M. Bauer; G. Greiner; T. Ertl, 2000. "Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization." In *SIGGRAPH/Eurographics Workshop on Graphics Hardware Conference Proceedings*. ACM SIGGRAPH, Addison Wesley, Interlaken, Switzerland, 109–118.

[31] Tegen, I.; A. A. Lacis, 1996. "Modeling of Particle Size Distribution and Its Influence on the Radiative Properties of Mineral Dust Aerosol." *Journal of Geophysical Research 101*, no. D14, 19237–19244.