View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

ADVANCED VISUALIZATION OF LARGE DATASETS FOR DISCRETE ELEMENT METHOD SIMULATIONS

Mark L. Sawley
LIN-ISE-STI, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

John Biddiscombe, Jean M. Favre
Swiss National Supercomputing Centre (CSCS), CH-6928 Manno, Switzerland

Abstract

State-of-the-art Discrete Element Method (DEM) simulations of granular flows produce large datasets that contain a wealth of information describing the time-dependent physical state of the particulate medium. To extract this information, both comprehensive and efficient post-processing methods are essential. Special attention must be paid to the interactive visualization of these large hybrid datasets containing both particle-based and surface-based data. In this paper, we report the use of the open-source visualization package *ParaView,* which we have customized specifically to perform advanced techniques for the post-treatment of large DEM datasets. Particular attention is given to the method used to render the individual particles, based either on triangulation of glyphs or using GPU-accelerated primitives. A demonstration of these techniques, and their relative merits when applied to the visualization of DEM datasets, is presented via their application to real industrial examples.

1. Introduction

Discrete Element Method (DEM) simulations are concerned with the time-varying interaction of a large number (potentially millions) of discrete particles with other particles and with boundary surfaces. Such simulations produce large datasets that are hybrid in nature, containing both particle-based (point) and mesh-based (surface) information. Post-processing of these datasets involves the application of both quantitative (generally involving statistical averaging) and qualitative (visualization) analyses to extract the required physical insights and information.

The present paper will be concerned exclusively with visualization methods. In particular, it is assumed that the user wishes to be able to probe the data interactively, and thus requires a sufficiently rapid response from the visualization system. Currently, a variety of different types of software are being used to visualize hybrid DEM datasets. These can be broadly categorized into the following groups:

1.1. Fully-integrated software

A number of commercial software packages have been developed over the years for the visualization of scientific data. Generally such data are expressed in terms of physical quantities discretized on a computational mesh ("mesh-based" data). These packages often provide a number of advanced features, and have a relatively short learning curve. Unfortunately, only a few of these software packages can be applied to datasets containing also point data that are not associated with a mesh ("particle-based" data). Particular examples of commercial software used for DEM simulations are *EnSight* [1] and *Tecplot* [2]. Since the basic functionality of these "black-box" software packages is generally not easily extended, such a fully-integrated approach is often overly limited for the visualization of DEM datasets.

1.2 Modular Visualization Environment (MVE) software

A more flexible approach is provided by MVEs, which provide a complete high-level programming environment that allows users to customize the software to their specific needs. Particular examples include *AVS/Express* [3] and *OpenDX* [4], the latter being a free open-source product based on software initially developed by IBM. Although generally more time-consuming to learn and implement, such software provide the flexibility and extendibility not offered by fully-integrated software. The cost of these advantages is generally reduced visualization performance, which can be a severe disadvantage for large DEM databases.

mark.sawley@epfl.ch; biddisco@cscs.ch; jfavre@cscs.ch

1.3 Library-based software

An approach that can provide flexibility while still maintaining performance is the use of "standard" computer graphics libraries. Library modules, which provide the basic functionality required by individual steps in the visualization chain, are encapsulated in a global environment. Particular examples of libraries used for the purpose of DEM visualization are *OpenGL* [5] and *VTK* [6]. Such an approach requires specific skills of the developer and generally a longer learning curve for the user. Nevertheless, when implemented in an appropriate manner, a library-based software approach provides the possibility of both flexibility and performance.

The present paper will concentrate on the use of a library-based software approach to visualize hybrid datasets, in particular the free open-source software *ParaView*. This widely-used package has been adapted for the specific visualization requirements of large DEM datasets.

2. ParaView

ParaView is a flexible, open-source, multi-platform software created by Kitware Inc. in conjunction with a number of US government laboratories [7]. It can be considered as a high-level interface for the *VTK* computer graphics library, which incorporates a wide variety of visualization algorithms and modelling techniques. *ParaView* includes a user-friendly interface (see Fig. 1) with the more commonly-used *VTK* library elements directly implemented as standard filters. This provides the basic functionality required for the visualization of DEM simulation data, such as independent display of particle and surface data, glyph representation of particles, colouring according to physical properties (e.g. particle velocity, size, density, surface forces), interactive view control and animation of time-varying datasets.

More advanced functionality can be incorporated into *ParaView* by accessing the appropriate *VTK* library element via a simple *XML* description. An example of particular interest is the creation of a continuum (meshed-based) representation of discrete (particle-based) data by 3D Delaunay triangulation; such a representation provides the possibility to use standard continuum visualization techniques (e.g. contour plots, cross-sectional cuts, streamlines).

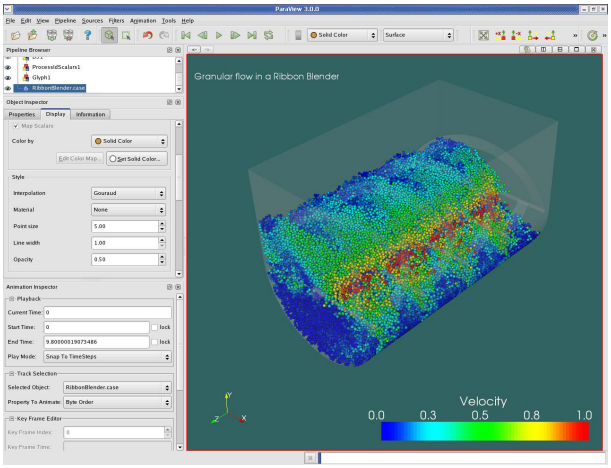

Figure 1 : ParaView graphical interface, with particles coloured according to their velocity.

For large DEM datasets with $O(10^5)$ particles or more, the visualization time can become prohibitive, even when using the latest generation of high-performance graphics workstations. This is particularly the case if a high-resolution glyph representation of the particles is used, since an excessively large number of triangles need to be rendered (see Fig. 2). Visualization would then consume such a vast amount of memory and overly large data transfer from the CPU to the GPU that rendering would be too slow to be useful for interactive purposes (see Table 1).

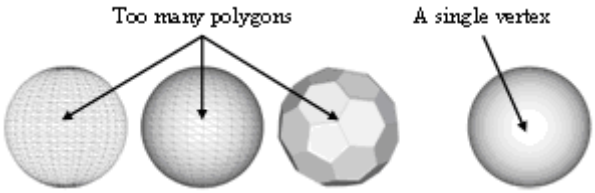

Figure 2 : Representation of particles using glyphs (left) and GPU primitives (right).

To overcome this problem two different approaches have been considered using *ParaView*: parallel computation and rendering, and the use of GPU-accelerated primitives for the particle representation.

2.1 Parallel computation and rendering

To facilitate the processing of large datasets, *ParaView* includes support for distributed parallel computation and rendering. When a large number of particles must be glyphed and rendered as spheres, distributing the work of both glyphing and rendering among nodes of a cluster is beneficial. As an example, the dataset with 100,000 particles shown in Fig. 1 has been loaded and rendered on 8

processors. Spherical glyphs with a low resolution $\{\theta,\phi\}$ of 8x8 when applied to 100,000 particles require 960,000 triangles, 500,000 points and a total memory of 475 MB. Parallel computation and rendering using *ParaView* allows the display of such large datasets and makes a previously unmanageable task possible.

# Particles	# Vertices / Sphere	Visual Quality	# Total Vertices	Total Memory	BW @ 25fps
Glyphs					
10^3	50	acceptable	50,000	600 kB	24 MB/s
10^3	100	good	100,000	1.2 MB	47 MB/s
10^6	50	acceptable	$50x10^6$	600 MB	24 GB/s
10^6	100	good	$100x10^6$	1.2 GB	47 GB/s
GPU					
10^6	1	excellent	$1x10^6$	12 MB	300 MB/s

Table 1 : Memory and bandwidth (BW) requirements for triangulated glyphs and GPU spheres.

2.2 GPU primitives for particle representation

An alternative approach that we have recently implemented into *ParaView* employs GPU-accelerated primitives to render particle-based data in a highly efficient manner. This technique has previously been used with great effect to render surfaces obtained from point sampling (e.g. laser acquisition of data) and also to render iso-surfaces generated from volume data (e.g. medical scans). The initial implementations used point splats defined as disk-like shapes coloured and shaded to match the underlying surface (see [8] for a thorough introduction). The technique has steadily evolved to consider higher curvature orders of the underlying surface, improved blending of overlapping splats, perspectively correct rendering of individually curved splats and in its latest form, the direct rendering of quadratic forms, or quadric surfaces [9].

The work of Sigg and co-workers [9] has been used as a basis for our implementation of GPU-rendered primitives. Their key development is to represent a quadratic form using a 4x4 homogeneous matrix. This allows a compact representation of an element (sphere, cylinder, cone, ellipsoid or even parabolic surface) as a 4x4 matrix that can be combined with the usual transform and lighting pipeline of graphics hardware. The sphere form is ideally suited to a DEM application and also can be highly optimized due to the simplicity of its representation. With a single $\{x,y,z\}$ point representing the centre of the sphere plus one parameter for the radius, a complete representation can be packed into a matrix, then transformed with correct perspective and lighting added (see Fig. 2). We have only implemented spheres to date, however, ellipsoids and other quadratic forms require only the reformulation of the matrix with additional parameters representing the polynomial terms in the equations of the quadratic functions. We also plan to extend our capabilities to represent rice-grain shaped objects by combining multiple cylinder / sphere / ellipsoid primitives into a single 'particle'. Table 1 indicates that the computer resources required for a GPU primitive representation of spherical particles are significantly less than for triangulated glyphs.

Our *ParaView* rendering extension supports effectively three modes of operation:

1. Drawing a texture to represent a particle (sprite)
2. Rendering of a sphere without depth / perspective
3. Rendering of an exact depth-correct sphere using full ray / sphere intersection testing

The advantage of mode 1 is speed. However, when zooming in closely on particles, the sprite image does not scale correctly; this mode is thus designed primarily for quick interaction. Mode 2 is also fast, but allows each particle to have an individually controlled radius as well as opacity and colour. The disadvantage of mode 2 is that, like mode 1, the particles are rendered as flat images and if they overlap either with each other, or with the mesh geometry, the images generated are incorrect. In general, DEM simulations treat particles as solid spheres with only a very limited overlap. In cases where particles do overlap, mode 3 provides a complete sphere evaluation performed by a ray intersection from the eye, through the particle, taking into account the thickness / depth of the particle correctly at each point. The particles may be mixed with scenery or other geometric objects without problem. Rendering using mode 3 is however an order of magnitude slower than for mode 2.

Table 2 shows the typical performance of particle rendering using GPU primitives. The render times

vary from one graphics card to another and improve whenever a new generation of graphics card is released. Rendering modes 2 and 3 use fragment shaders to perform the evaluation of the sphere at run time, which also makes them sensitive to the size of the screen space occupied by spheres. Many large highly-magnified particles will render more slowly than the same particles at a distance due to the total number of pixels for which the evaluation is performed. Timing comparisons are therefore subject to a number of parameters and the numbers presented in Table 2 represent a simple 'average' during typical daily usage.

Rendering Method	Millions of particles per second
Raw vertex	50
Sprite texture (mode 1)	40
Simple sphere (mode 2)	30
3D exact sphere (mode 3)	1

Table 2 : Render times for particles using GPU spheres/textures/points (using a NVidia GeForce 7 series graphics card).

Using our implementation of GPU primitives in *ParaView* it is possible to visualize datasets with the order of one million particles at interactive rates on a standard desktop workstation. Making use of parallel rendering capabilities gives us the ability to render easily datasets with the order of tens of millions of particles on a visualization cluster, while considering hundreds of millions of particles would be possible should the need arise. Currently the pace of GPU improvement outstrips the pace of DEM simulation growth and we have not yet reached the size limit.

3. DEM simulations

The simulation data analyzed in the present paper were produced by a state-of-the-art 3D DEM solver developed by Granulair Technologies [10]. This software incorporates diverse physical and numerical modelling that enables a wide range of problems of industrial and academic interest to be addressed.

The solver considers particles of either spherical or non-spherical (comprised of a cluster of spherical particles) shape, with any distribution of particle properties (e.g. size, density). The boundary surfaces are treated as a collection of objects, with each object composed of triangular and/or quadrilateral elements produced by standard mesh-generation software. Objects can be in relative translational, rotational or vibrational motion. The influence of an interstitial fluid is modelled by either considering the single-particle fluid drag, or by coupling the DEM solver to a Navier-Stokes (CFD) solver using an effective porous medium approach [11]. Cohesive forces between particles and with boundary surfaces can also be treated to model the influence of liquid bridge formation and other attractive forces.

A soft-particle model is employed by the DEM solver, with the normal and tangential collisional forces depending on the particle overlap. The inter-particle and particle-surface interactions are detected using a two-step process involving spatial sorting to produce a nearest neighbour list followed by explicit contact detection. A spring-dashpot model is used to model the collisional forces [12]. The governing equations expressing the conservation of linear and angular momentum are resolved using an explicit time-integration scheme.

Various types of data are exported during a DEM simulation for both qualitative and quantitative analysis. For the present study, the data to be visualized is exported in EnSight 6 format [1], which is directly readable by *ParaView*.

To illustrate the visualization methods presented in this paper, three different problems of industrial interest are considered. A description of each problem and the DEM simulations undertaken is briefly presented.

3.1 Ribbon blender

The ribbon blender is a convective mixer used in numerous industrial sectors (e.g. pharmaceuticals, foods, chemicals, plastics) to mix particulate material. The ribbon blender considered here consists of two helical ribbons of opposite pitch that rotate in a static cylindrical trough. The trough has a length of 500 mm and a width of 300 mm. The ribbons of width 30 mm rotate at a speed of 30 rpm. DEM simulations have been undertaken for a charge of 100,000 identical spherical particles of diameter 6 mm. Figure 3 presents the computational mesh used to discretize the mixer geometry, which consists of both triangular and quadrilateral elements selected to describe in an optimal manner the different surfaces of the mixer.

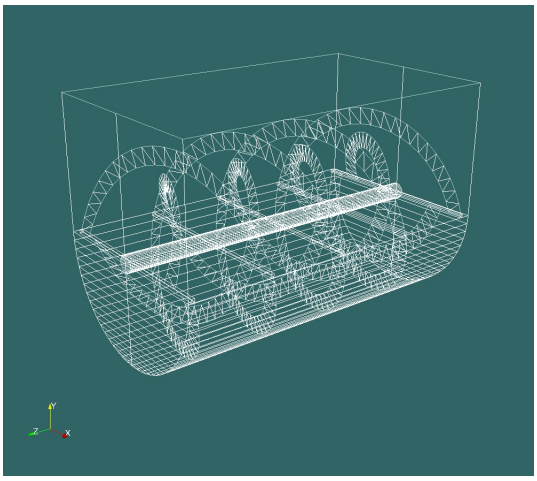

Figure 3 : Discretization of the surfaces of the ribbon blender geometry.

3.2 Conveyor loading and transfer

The loading of a belt conveyor and the transfer of material from one moving belt conveyor to another are important bulk handling tasks. Such operations are widely used for the transport of both coarse grain material (such as cereals, mineral ore and coal) and fine powders (as used in the chemical and pharmaceutical industries). The dual belt conveyor system shown in Fig. 4 is considered here. The granular material initially stored in a rectangular hopper is loaded onto the upper belt using a feed chute with lateral skirts. The material is transferred to the lower belt at 90^{o} via a transfer chute, comprised of a box containing two curved impact plates. Both upper and lower belts have a width of 450 mm and speed of 2.5 m/s. The granular material consists of (initially 25,000) spherical particles with diameters distributed between 15 and 30 mm, a density of 2500 kg/m^3 and a flow rate corresponding to 47 t/hr.

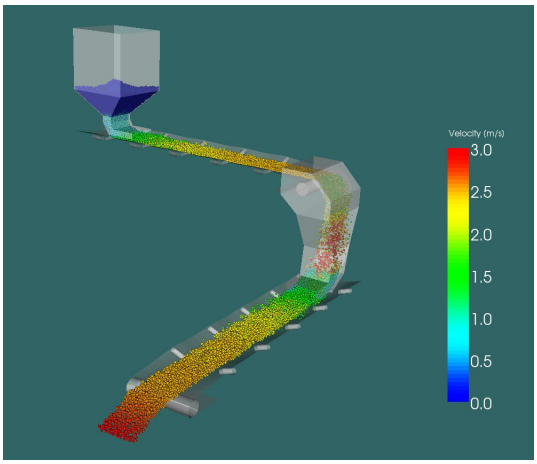

Figure 4 : Particulate flow on a dual belt conveyor system with a transfer chute.

3.2 Sprouted bed

Sprouted beds are used in industry for a variety of tasks including drying, coating and enhancing chemical reactions. In a sprouted bed, air enters into a vertical hopper through a nozzle above which the granular material is fluidized. The particles begin to circulate, falling into the surrounding area of low-velocity air, before re-entering the air stream. The numerical simulation of such a multiphase system can be undertaken using a coupled CFD-DEM approach. The commercial CFD solver FLUENT 6 was used to obtain the air flow results. For the present illustrative purposes, a 2D flow simulation was performed using 5000 identical 3 mm diameter spheres of density 2500 kg/m^3 in a hopper of total width 152 mm and nozzle width 19 mm.

4. Visualization examples

ParaView has been employed for the visualization of DEM datasets corresponding to a wide variety of different applications. In the present section, some representative results are presented for the above-mentioned applications.

4.1 Continuum representation

As indicated in Section 2, it can be useful for analysis purposes to convert the discrete particle representation generated by the DEM solver to a continuum representation. To illustrate the possibilities provided by such an approach, we consider the specific example of granular mixing in the ribbon blender.

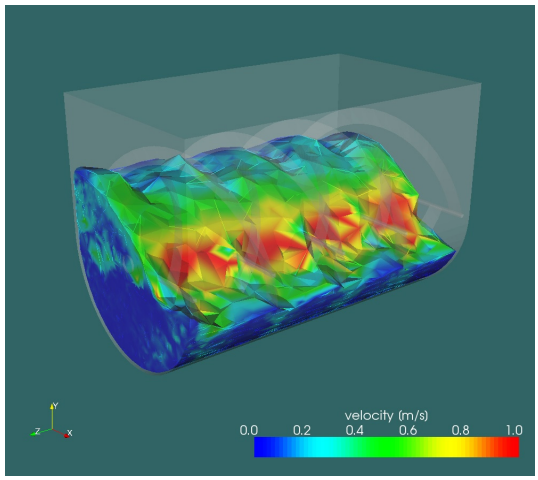

Figure 5 : Delaunay triangulation of the particle-based dataset.

The Delaunay triangulation of the particle-based dataset was performed using the `vtkDelaunay3D`

class contained in the *VTK* library. This results in a continuous medium filling the lower portion of the trough up to the free surface of the flow, as shown in Fig. 5. It should be noted that 3D Delaunay triangulation is a time-consuming task; in *ParaView* the computation can be accelerated using parallel computing.

Based on this continuum representation of the DEM data, different standard visualization techniques can be applied. As an example, Fig. 6 shows the velocity contours on three cutting planes at different axial positions, using the `vtkCutter` class implemented in *ParaView*. These contours indicate in a clear manner, the variation of the particle avalanching along the free surface, as well as the influence of the mixer ribbons on the local flow velocity.

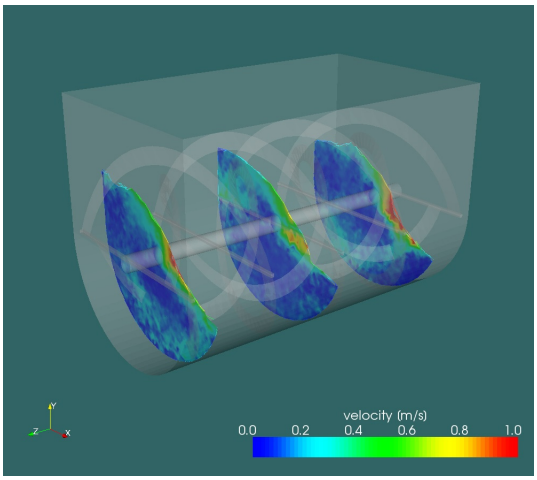

Figure 6 : Velocity contours on three cutting plane at different axial locations.

4.2 Streamlines and pathlines

It is often useful to trace the movement of individual particles in a granular flow. At a given time, a *streamline* is defined as the curve that is tangent to the (instantaneous) velocity field at every point. A *pathline* is defined as the trajectory a particle follows when released into a (time-varying) flow field. Therefore, whereas a streamline indicates the trajectory of a fictive massless particle injected into the (stationary) flow field, a pathline shows the actual trajectory taken by a physical particle.

Streamlines can be computed in a straightforward manner from the continuum representation of the DEM data using the `vtkStreamLine` class implemented in *ParaView*. An example for the ribbon blender is shown in Figure 7. The plotted line represents the path that a fictive massless particle would take in the velocity field calculated at that particular instant of time. By plotting streamlines originating at different spatial locations, physical phenomena of interest – such as the existence of regions of flow recirculation – can be elucidated.

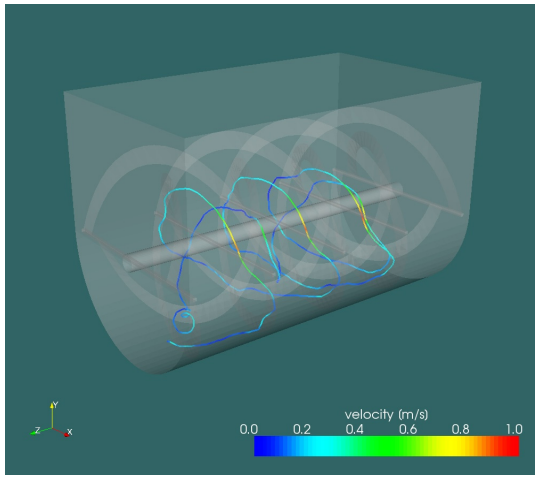

Figure 7 : A streamline through the velocity field.

Pathlines are calculated in *ParaView* by tracing the time-evolution of the positions of selected particles. Figure 8 shows the pathlines calculated for particles in the ribbon blender, while Fig. 9 shows the pathlines of particles being loaded onto a conveyor belt. The latter example shows that some particles bounce off the conveyor belt and eventually leave the computational domain. The use of interactive 3D viewing of the pathlines enables a detailed inspection of particle trajectories, and thus provides insights into the behaviour of individual particles. In addition, plotting pathlines has proven to be an invaluable means of debugging simulations to identify erroneous motion introduced by potential errors in the DEM simulation code.

Pathline plotting is only possible if one of two conditions is met by the data:

- the number of particles is invariant throughout the simulation and the order in which data are written is the same for all time steps,
- each particle has a unique identifier which remains unchanged throughout the simulation.

If either of these conditions is not met, it is not possible to track particles unambiguously.

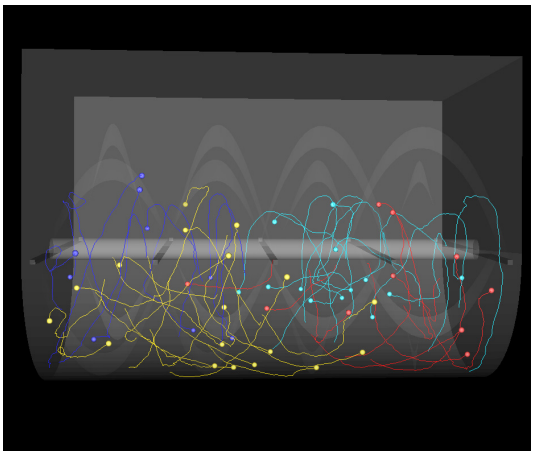

Figure 8 : Pathlines of particles in the ribbon blender.

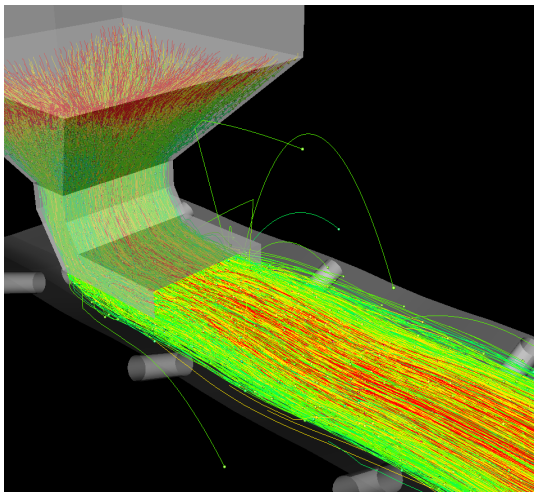

Figure 9 : Pathlines for particle loading onto a conveyor belt.

4.3 Transparency rendering

When DEM simulations are performed for dense granular flows (as is the case for the ribbon blender example), it can be difficult to observe the behaviour of particles within the flow. While cutting planes can be used to reveal internal details, this has the drawback of hiding features that may be desired for contrast or comparison. In some cases, volume rendering of the triangulated mesh data will produce good results, with the opacity of the rendered image being determined by a physical scalar quantity.

Volume rendering is supported by *ParaView* but requires a triangulation of the particles. As mentioned above, Delaunay triangulation is time-consuming and therefore not well suited to interactive animation. With our custom renderer, we can achieve the same effect by providing an opacity value, computed from a physical scalar quantity, for each particle and performing a depth sort on the fly during rendering.

Figure 10 illustrates an example of transparency rendering for flow in the ribbon blender. In this figure, a clear indication of regions of high particle spin (due to the motion of the helical ribbons) is revealed in the volume of the granular material.

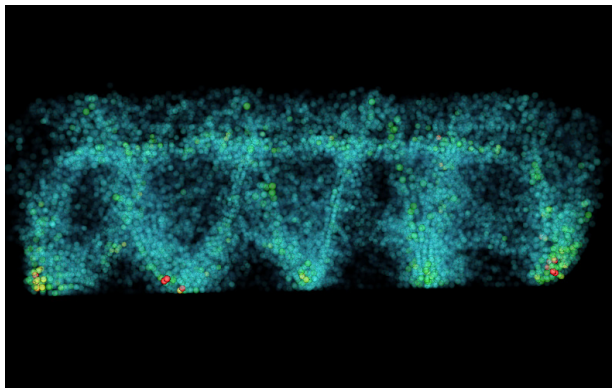

Figure 10 : Transparency rendering of particles to highlight a chosen physical quantity (particle spin in this example).

In the present implementation, sorting of the particles prior to rendering is performed on the CPU, and reduces rendering speed by a factor of around 5 (depending on the number of particles). However, it is planned to perform the sort on the GPU in the near future, which will accelerate the process significantly.

4.4 Multiphase fluid-granular flow

Multiphase flows are of particular importance in many industrial processes, and provide a wealth of detailed physical behaviour of particular interest. In order to extract the maximum amount of information from a coupled CFD-DEM simulation, it is essential to be able to visualize the computed results in an appropriate manner. The ability to superimpose different visual results produced from different input datasets is essential in this context.

An example of the visualization of coupled fluid-granular flow is presented in Fig. 11 for the sprouted bed. To view simultaneously the fluid and granular flows, transparency is applied to the continuum fluid results. This figure shows clearly the interaction between the granular material and the fluid flow, which results in the production of a characteristic fountain at the centre of the channel.

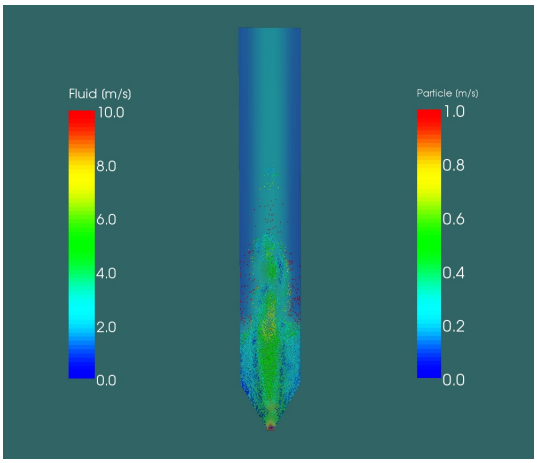

Figure 11 : Coupled fluid-granular flow in a sprouted bed (coloured by velocity).

5. Conclusions

The present study has presented a library-based software approach to the visualization of DEM simulation results. It has been demonstrated that such an approach provides both the flexibility and performance required to enable detailed interactive visualization of large hybrid datasets.

A number of novel methods, implemented into the widely-used open-source software *ParaView*, have been illustrated in this paper. Contrary to conventional triangulated glyphs, the use of GPU-accelerated primitives for the particle representation enables the interactive rendering of the order of one million particles on a standard desktop workstation. Delaunay triangulation of the 3D particle data produces a representation that can be used as a basis for standard continuum visualization techniques (e.g. contour plots, cross-sectional cuts, streamlines). Transparency rendering enables features hidden within dense granular flows to be elucidated.

The application of these advanced visualization techniques opens new avenues in the analysis of DEM simulation results for large-scale industrial problems.

Acknowledgements

The authors wish to acknowledge Sébastien Wiederseiner for providing the DEM simulation data for the sprouted bed.

References

[1] Computational Engineering International (CEI); see www.ensight.com

[2] Tecplot Inc.; see www.tecplot.com

[3] Advanced Visual Systems (AVS); see www.avs.com

[4] OpenDX; see www.opendx.org

[5] OpenGL; see www.opengl.org

[6] *The VTK User's Guide*, Kitware Inc. 2006; see www.vtk.org

[7] A.H. Squillacote, *The ParaView Guide: A Parallel Visualization Application*, Kitware Inc. 2006; see www.paraview.org

[8] M. Zwicker, H. Pfister, J. van Baar, M. Gross, *Surface Splatting*, SIGGRAPH 2001.

[9] Ch. Sigg, T. Weyrich, M. Botsch, M. Gross, *GPU-based ray-casting of quadratic surfaces,* Eurographics Symposium on Point-Based Graphics 2006, pp. 59-65; see graphics.ethz.ch/~mbotsch/publications/pbg06.pdf

[10] Granulair Technologies; see www.granulair.com

[11] Y. Tsuji, T. Kawaguchi, T. Tanaka, *Discrete particle simulation of two-dimensional fluidized bed*, Powder Technology, **77**, 79-87 (1993).

[12] P.A. Cundall, O.D.L. Strack, *A discrete numerical model for granular assemblies*, Géotechnique, **29**, 47-65 (1979).