An Efficient Linear Octree-Based Grid Toward Magnetic Fluid Simulation

Sean Flynn
Brigham Young University

Parris Egbert
Brigham Young University

ABSTRACT

An Eulerian approach to fluid flow provides an efficient, stable paradigm for realistic fluid simulation. However, its reliance on a fixed-resolution grid is not ideal for simulations that exhibit both large and small-scale fluid phenomena. A coarse grid can efficiently capture large-scale effects like ocean waves, but lacks the resolution needed for small-scale phenomena like droplets. On the other hand, a fine grid can capture these small-scale effects, but is inefficient for large-scale phenomena. Magnetic fluid, or ferrofluid, illustrates this problem with its very fine detail centered about a magnet and lack of detail elsewhere. Our new fluid simulation technique builds upon previous octree-based methods by simulating on a custom linear octree-based grid structure. A linear octree is stored contiguously in memory rather than as a recursive set of pointers. This use of memory improves the cache coherency issues inherent in previous octree methods. By localizing high-resolution regions we can allow the simulation of small-scale phenomena, while at the same time maintaining efficiency in coarse grid regions. We believe our new simulation technique will provide a framework for simulating ferrofluids which have not been simulated in a physically-based manner from a computer graphics perspective as of yet.

KEYWORDS

fluid, simulation, physically-based animation, octree, adaptive refinement, magnetic fluid, ferrofluid

1 INTRODUCTION

Realistic visual fluid simulation research is increasingly important as high-performace physical simulation work becomes more prevalent in a variety of fields including animation, video games, science, and engineering. Over the past few decades, computer graphics research has had great success in creating convincing fluid phenomena including ocean waves, smoke plumes, water splashing into containers, etc. By taking a physically-based approach, the burden on artists needing to hand-animate complex fluid behavior has been alleviated. Additionally, by providing controllable parameters, modern fluid simulation techniques provide artists some control over simulations. However, because physically-based fluid simulation is computationally intensive, efficient algorithms and data structures are crucial to support the ever-increasing demand for new types of simulations. Depending on the art direction of a simulation, the simulator's configuration details can vary widely. For example, simulating large-scale ocean waves at the resolution needed to accurately resolve tiny droplets would be very inefficient, and attempting to simulate droplets with a very coarse resolution would not have the necessary accuracy. This is a problem when a fluid simulation must simultaneously capture large and small-scale phenomena. A particularly interesting fluid that illustrates this problem is ferrofluid.

Ferrofluid, which was invented in 1963 by NASA's Steve Papell [18], is a liquid that carries suspended ferromagnetic nanoparticles which allow it to become strongly magnetized in the presence of a magnetic field. When magnetized, ferrofluid exhibits an interesting spiking pattern centered about the magnet. This spiking phenomenon is shown in figure 1. These spikes are a result of the balance between the forces due to gravity, surface tension, and the magnetic field. Although there have been recent attempts at visually simulating ferrofluid, there is currently no efficient physcially-based solution that captures all of its features. Because of its many applications, which range from loudspeaker cooling to drug targeting, an efficient physically-based simulation system would provide a versatile, low-cost means of testing for ongoing research efforts in a variety of fields.

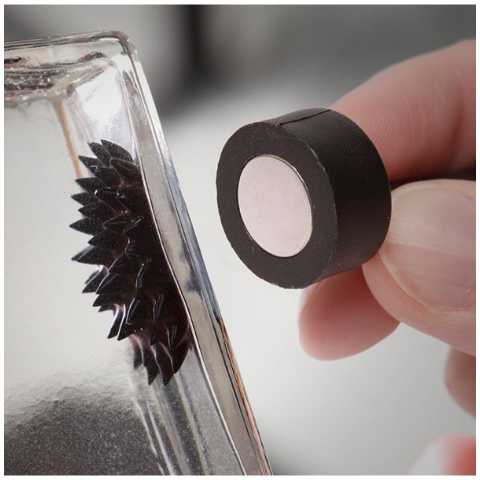


Figure 1: Ferrofluid exhibits an interesting spiking behavior when in the presence of a magnetic field.

While ferrofluid reacts and attracts to the magnet at a macro level, it only exhibits highly detailed spikes very close to the magnet. As the magnet gets closer to the ferrofluid, the spikes become smaller and more frequent. Thus there is a large disparity between the level of detail needed in a ferrofluid simulation; a low resolution is adequate for the macro features, but a very high resolution is needed to accurately compute the spikes. Our new linear octree grid structure caters to these types of simulations by allowing varying resolutions based on the specific simulation requirements.

To numerically solve the Naviér Stokes equations and realistically simulate fluid flow there are two primary approaches: Eulerian and Lagrangian simulation. An Eulerian approach uses a three-dimensional grid that tracks the properties of the fluid like velocity and pressure at fixed locations that do not move. The fluid flows through the grid in an Eulerian simulation. On the other hand, a Lagrangian simulation tracks particles or quantities of fluid as they move and interact with one another. The fluid properties are attached to the moving particles without the use of a grid. Our approach is primarily Eulerian due to its use of space discretization on a grid.

With a traditional fixed uniform grid the user must specify the resolution of the grid before simulating. Because increasing resolution has a large impact on the simulation time, the lowest possible resolution that still achieves the desired effect will ideally be specified. As previously mentioned, if the simulation requires both large and small details, a grid with fixed resolution will be either inefficient or inaccurate. Our approach allows regions of the Eulerian simulation grid to be subdivided based on the simulation requirements with the use of an octree based-structure. Our approach also alleviates the cache coherency issues inherent in previous octree-based methods by storing the octree linearly in memory rather than as a pointer-based tree structure.

2 PREVIOUS WORK

Modern fluid dynamics from a computer graphics perspective began in 1996 when Foster and Metaxas presented a comprehensive Eulerian fluid simulator [5]. Their work was mainly based on an earlier computational physics paper [6]. Jos Stam improved upon this work while attempting to create a production fluid simulator [17]. Rather than relying on explicit Eulerian schemes which required small timesteps and suffered from instability, Stam developed a method that made the simulation unconditionally stable for the first time. This was done by adding a Lagrangian technique now commonly known as the backward particle trace. These seminal papers revolutionized visual fluid simulation by efficiently approximating the Naviér Stokes equations and allowed for a wide range of fluid effects without the need for specific logic.

Previous approaches to free-surface tracking were mostly done with either marker particles or with height-field techniques. While these approaches provided a general approximation of liquid surfaces, they suffered from mass dissipation. Foster and Fedkiw presented a method that used level sets to accurately track the fluid surface while avoiding mass dissipation [4]. This level set method approach was improved by Enright, Marschner, and Fedkiw by using particles on both sides of the free surface to perform a thickening step [3]. This improvement shifted the liquid rendering paradigm from a volumetric approach to a surface based approach which allowed for more convincing ray tracing techniques to be used.

Lagrangian approaches to fluid simulation are also insightful as our simulation approach has Lagrangian elements. Smoothed-particle hydrodynamics (SPH), which was initially introduced by Desbrun and Gascuel [2] and then adapted for efficient computer graphics simulations by Muller, Charypar, and Gross [11], is a commonly used Lagrangian method. The simulation technique we use in this work is a hybrid Eulerian-Lagrangian technique based on the Fluid-Implicit Particle method (FLIP) originally developed by Brackbill [1] and adapted for computer graphics by Zhu and Bridson [22]. FLIP provides the benefits of performing computation on a fixed grid, but by storing and advecting the velocity field on particles rather than the grid, numerical averaging is reduced and the resulting simulations exhibit less unintentional viscosity.

Any fluid simulation technique that uses a grid has the potential to benefit from improvements to the design of the grid. This is obvious in pure Eulerian approaches, but also applies to a variety of hybrid techniques including FLIP. Additionally, many modern fluid simulators use multiple grids for ancillary computations like fluid surface remeshing [21], or surface tension computation [20]. Our research builds upon a number of previous adaptive grid techniques which will now be described.

Losasso, Gibou, and Fedkiw presented an alternative to uniform grids by performing fluid simulation on an adaptive octree grid [10]. They built upon a previous octree method [14] by extending it to use unrestricted octrees and adding support for free-surface computation. They were able to reformulate the pressure solve and maintain a symmetric positive definite discretization. However, this approach is known to suffer from instability due to oscillatory spurious velocities at coarse-to-fine interfaces within the octree. This instability was improved by Olshanskii, Terekhov, and Vassilevski with the use of a low-pass filter [13]. Our approach is based on an earlier octree-based approach which used a less efficient, but more stable hierarchical pressure solve [16]. Octrees provided improved efficiency for large-scale simulations with varying levels of detail. However, due to the fragmented memory layout of a tree structure in addition to the overhead of complicated pointer logic, these improvements were limited.

There has been little work done from a computer graphics perspective on simulating magnetic phenomena. In 2012 Thomaszewski presented a system for simulating magnetic rigid bodies [19]. Recently, magnetic fluids were simulated from a visual perspective using procedural approaches [7, 8]. However, as far as we are aware, no physically-based method has been developed to accurately simulate ferrofluid in the computer graphics community.

3 METHODOLOGY

At a high level, our fluid simulation technique in its current state consists of the following two steps:

(1) Grid configuration
   - The user defines the simulation parameters, the global resolution of the linear octree, and subdivides any regions that require additional detail.

(2) Simulation
   - Our simulation method closely follows the previously mentioned octree methods. Specifically, we use a hierarchical pressure solve.

3.1 Linear Octree Structure

We now describe the data structure our approach relies on as it is the primary contribution of this paper. An octree data structure, first described in [9], is an efficient partitioning of a three-dimensional space. It is defined by recursively subdiving each cell into eight octants. Starting with the root node, each node has a pointer to each of its eight children down to the leaf nodes, which do not point to any nodes. Each cell is subdivided as far as desired based on the required detail. A two-dimensional cross section of this is shown in figure 2.

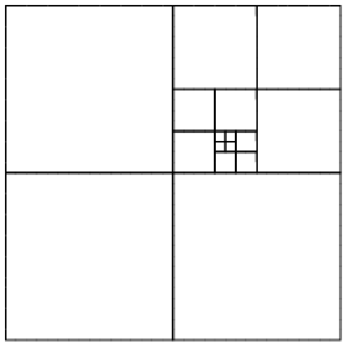


Figure 2: A two-dimensional cross section of an octree.

Our simulation method uses a custom linear octree. A linear octree shares the same interface as a traditional octree, but it is stored in a single contiguous chunk of memory rather than being a recursive set of pointers.

This is done by predefining the minimum and maximum cell size and then fully building out the structure sequentially in memory using those sizes. This has the disadvantage of always maximizing memory usage, but it allows the size of the tree to remain constant and be stored contiguously. The primary advantage of this linear structure is that the cache coherency issues inherent in previous octree simulators are alleviated. Rather than needing to deallocate and allocate memory at each time step as with a traditional octree, our method works by simply setting parameters on the cells which indicate their respective sizes. While this will use more memory, it allows us to improve accuracy and speed which is the focus of our work.

The implementation details of our linear octree grid structure will now be described. To alleviate the need to use floating point numbers for the sizes of the cells in the octree, the minimum cell size is always set to one. The maximum cell size is then any power of two. Figure 3 shows how the linear octree grid is stored in memory as well as the order in which iteration takes place.

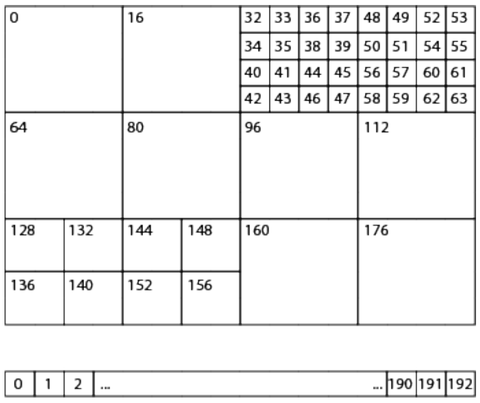
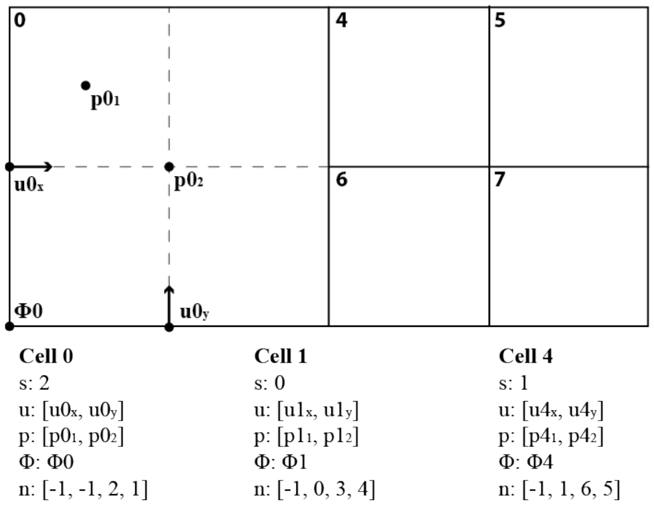


Figure 3: A two-dimensional example of our linear octree structure. The labels show the linear array indices. This ordering simplifies iterating over the grid. In this two-dimensional case the minimum cell width is 1 and the maximum is 4. The size of cells 0, 32, and 128 would be 4, 1, and 2 respectively.

Each cell stores its size *s*, a vector *u* representing each component of velocity, a list of pressure values, a signed distance value ϕ, and the indices of each of its eight neighbors in the grid. The size determines how the cell will be treated as if it were part of a traditional octree. A cell with size 0 indicates it is part of a larger, less subdivided cell. Each of the x, y, and z components of the velocity vector are stored at the minimal cell face centers as with traditional Marker-and-Cell (MAC) grids. A list of pressure values is used rather than a single pressure value because we solve for pressure hierarchically. This will be described in greater detail in the next section. The signed distance value ϕ, stored at the cell corner, indicates the distance to the fluid surface. Finally, each cell stores the indices of each of its neighbors to make neighbor lookups easier. Figure 4 illustrates the data contained in each cell.



Cell 0	Cell 1	Cell 4
s: 2	s: 0	s: 1
u: [$u0_x$, $u0_y$]	u: [$u1_x$, $u1_y$]	u: [$u4_x$, $u4_y$]
p: [$p0_1$, $p0_2$]	p: [$p1_1$, $p1_2$]	p: [$p4_1$, $p4_2$]
Φ: Φ0	Φ: Φ1	Φ: Φ4
n: [-1, -1, 2, 1]	n: [-1, 0, 3, 4]	n: [-1, 1, 6, 5]

Figure 4: Sample cells in our linear octree structure. The grid is shown in two dimensions for clarity. Shown below the diagram is data from cell 0, 1, and 4. In the neighbors list *n*, -1 values indicate no neighbor in that direction. Cell 1 is size 0 and so it is effectively ignored until cell 0 is subdivided. The dotted line indicates that there are additional cells stored in memory which are ignored because their size is 0.

Our custom implementation of the linear octree provides the interface necessary to utilize previous octree simulation algorithms, but with the caching benefits of a traditional uniform grid. It also reduces the overhead in traversing up and down a tree structure to find neighbors because they are easily computed due to the predictable memory arrangement.

3.2 Simulation

Our simulation technique draws upon the previously mentioned hierarchical pressure solve [16], but adapts it to free-surface fluids like water using the level set techniques presented in the more recent octree fluid simulation paper [10].

Fluid dynamics are governed by the Naviér Stokes equations. For computer graphics we use the incompressible version of these equations because for fluids like water, compressibility is visually negligible. These equations are as follows:

$$\nabla \cdot \mathbf{u} = \mathbf{0} \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \gamma \nabla^2 \mathbf{u} + \mathbf{F} \tag{2}$$

Equation 1 is the incompressiblity term. That is, the amount of fluid moving into and out of a volume must be the same.

Equation 2 describes the motion of the fluid. The change in velocity is a result of a number of forces. First is the velocity convection term, or how the fluid is moving about itself due to the conservation of momentum. The pressure gradient is subtracted to model the fact that fluid always moves from regions of higher pressure to lower pressure. A viscosity, or internal friction force is then added, and finally external forces like gravity or collisions are added.

We will now describe our hierarchical pressure solve. The key idea is to solve for pressure as if the grid were uniform at each of the subdivision levels present in the grid. This way the standard uniform pressure solve algorithm described in the previous works can be used. First, we iterate over the cells of the grid at the largest size and when a cell is encountered that is subdivided, the average of the values stored at the child nodes is used. With pressure now defined at the lowest resolution, we proceed to solving at the next smallest size. The second solve is just another uniform pressure solve, but with cells that are smaller. Again, if cells are further subdivided, the average of the child values is used. This process is repeated until the pressure is solved from the lowest to the highest resolution. With each solve, the region of the grid being considered becomes smaller and smaller, and the time spent solving for pressure is focused on the regions of the grid that are subdivided the most.

3.3 Simulating Ferrofluid

To this point we have only discussed simulating traditional fluid flow. While our system has not matured to the point where it is able to begin simulating magnetic fluids, we have a number of promising directions to pursue.

There are three primary components to the spiking behavior of ferrofluid: surface tension, the magnetic force, and gravity. Because our grid can be efficiently subdivided to an arbitrary resolution, we should be able to attain enough accuracy to delicately balance these three forces to achieve the spiking pattern. The resolution of the grid will be set to dynamically subdivide near the location of the magnet. This way we can ensure the highest resolution is used only where it is needed.

Initially we plan to implement surface tension as a modification to the pressure solve based on the curvature similarly to how it is implemented in the previous octree approach [10]. The curvature is simply defined as the laplacian of the signed distance field at the fluid-surface interface, or zero level set. The issue that may arise using this method is that even at a very high resolution, level set methods tend to wash away fine details, and the very fine spiking shape may be difficult to stably maintain. If this is the case, there are other non-level set surface tension methods we might try.

Wojtan and Thurey recently developed a triangle-based free-surface computation technique [20, 21]. Their technique allows for very fine detail preservation while simplifying the mesh in areas where detail is not needed. The surface tension effects achieved by this method are impressive. This technique relies on the resolution of ancillary grids that could take advantage of our new grid structure. Their approach could potentially be a better fit for the accuracy needed for ferrofluid simulation.

With surface tension accurately modeled, and after trivially adding in the gravity force, the final step is to model the magnetic force. We plan to implement the technique outlined in the magnetic rigid body paper [19]. This will require adapting their approach to fluids. Additionally, we plan to look into techniques used by the computational physics and engineering communities in simulating ferrofluid [12, 15]

4 CURRENT PROGRESS AND FUTURE WORK

The implementation of our system is an ongoing effort. Our initial test case is a simulation of a crown splash. Crown splashes demonstrate that the simulation can effectively model surface tension. Figure 5 shows a simple crown splash in two-dimensions on our grid without any subdivision added. Once our linear octree grid is fully implemented, we will recreate this test with various levels of subdivision. We will record the runtimes and memory utilization of our system and compare them to the same simulation setup on a uniform simulation grid, and on a traditional pointer-based octree grid. Initial tests are promising but are inconclusive as our data structure is not yet complete.

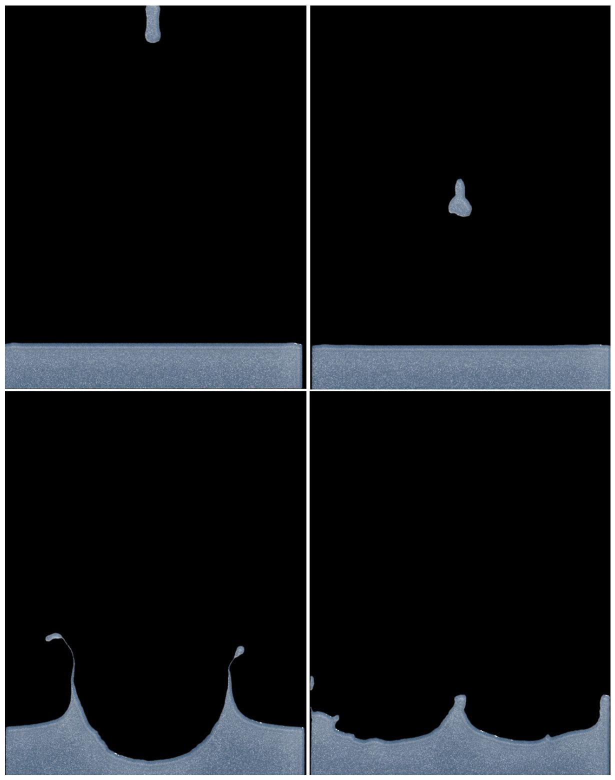


Figure 5: A simple two-dimensional simulation of a droplet of liquid falling and forming a crown splash as it hits a resting body of liquid on a uniform simulation grid. Crown splashes are a good test case of surface tension.

With our linear octree grid fully implemented and tested, we will move onto implementing and testing the magnetic force techniques described in the previous section. We may also look into automating or improving the process of subdividing the grid. It might be useful if the grid automatically subdivides based on the position of an object, like a magnet in the case of ferrofluid.

REFERENCES

[1] J U Brackbill and H M Ruppel. 1986. FLIP: A Method for Adaptively Zoned, Particle-in-cell Calculations of Fluid Flows in Two Dimensions. *J. Comput. Phys.* 65, 2 (Aug. 1986), 314–343. DOI:http://dx.doi.org/10.1016/0021-9991(86)90211-1

[2] Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*. Springer-Verlag New York, Inc., New York, NY, USA, 61–76. http://dl.acm.org/citation.cfm?id=274976.274981

[3] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. 2002. Animation and Rendering of Complex Water Surfaces. *ACM Trans. Graph.* 21, 3 (July 2002), 736–744. DOI:http://dx.doi.org/10.1145/566654.566645

[4] Nick Foster and Ronald Fedkiw. 2001. Practical Animation of Liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 23–30. DOI:http://dx.doi.org/10.1145/383259.383261

[5] Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graph. Models Image Process.* 58, 5 (Sept. 1996), 471–483. DOI:http://dx.doi.org/10.1006/gmip.1996.0039

[6] Francis H. Harlow and J. Eddie Welch. 1965. Numerical Calculation of TimeâĂŘDependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids* 8, 12 (1965), 2182–2189. DOI:http://dx.doi.org/10.1063/1.1761178

[7] Tomokazu Ishikawa, Yonghao Yue, Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. 2012. VISUAL SIMULATION OF MAGNETIC FLUIDS. In *Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications (VISIGRAPP 2012)*. 319–327. DOI:http://dx.doi.org/10.5220/0003867303190327

[8] Tomokazu Ishikawa, Yonghao Yue, Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. 2013. *Visual Simulation of Magnetic Fluid Using a Procedural Approach for Spikes Shape.* Springer Berlin Heidelberg, Berlin, Heidelberg, 112–126. DOI:http://dx.doi.org/10.1007/978-3-642-38241-3_8

[9] Rensselaer Polytechnic Institute. Image Processing Laboratory and D.J.R. Meagher. 1980. *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer.* https://books.google.com/books?id=CgRPOAAACAAJ

[10] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. ACM, New York, NY, USA, 457–462. DOI:http://dx.doi.org/10.1145/1186562.1015745

[11] Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159. http://dl.acm.org/citation.cfm?id=846276.846298

[12] Curtis M. Oldenburg, Sharon E. Borglin, and George J. Moridis. 2000. Numerical Simulation of Ferrofluid Flow for Subsurface Environmental Engineering Applications. *Transport in Porous Media* 38, 3 (2000), 319–344. DOI: http://dx.doi.org/10.1023/A:1006611702281

[13] Maxim A. Olshanskii, Kirill M. Terekhov, and Yuri V. Vassilevski. 2013. An octree-based solver for the incompressible Navier-Stokes equations with enhanced stability and low dissipation. (2013).

[14] Stéphane Popinet. 2003. Gerris: A Tree-based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries. *J. Comput. Phys.* 190, 2 (Sept. 2003), 572–600. DOI: http://dx.doi.org/10.1016/S0021-9991(03)00298-5

[15] Dongxiao Shi, Qincheng Bi, and Rongqi Zhou. 2014. Numerical Simulation of a Falling Ferrofluid Droplet in a Uniform Magnetic Field by the VOSET Method. *Numerical Heat Transfer, Part A: Applications* 66, 2 (2014), 144–164. DOI: http://dx.doi.org/10.1080/10407782.2013.869459 arXiv:http://dx.doi.org/10.1080/10407782.2013.869459

[16] Lin Shi and Yizhou Yu. 2002. *Visual Smoke Simulation with Adaptive Octree Refinement.* Technical Report. Champaign, IL, USA.

[17] Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128. DOI: http://dx.doi.org/10.1145/311535.311548

[18] P.S. Stephen. 1965. Low viscosity magnetic fluid obtained by the colloidal suspension of magnetic particles. (Nov. 2 1965). https://www.google.com/patents/US3215572 US Patent 3,215,572.

[19] Bernhard Thomaszewski, Andreas Gumann, Simon Pabst, and Wolfgang Strasser. 2008. Magnets in Motion. *ACM Trans. Graph.* 27, 5, Article 162 (Dec. 2008), 9 pages. DOI: http://dx.doi.org/10.1145/1409060.1409115

[20] Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. 2010. A Multiscale Approach to Mesh-based Surface Tension Flows. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 29, 3 (2010).

[21] Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. 2009. Deforming Meshes That Split and Merge. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09)*. ACM, New York, NY, USA, Article 76, 10 pages. DOI: http://dx.doi.org/10.1145/1576246.1531382

[22] Yongning Zhu and Robert Bridson. 2005. Animating Sand As a Fluid. *ACM Trans. Graph.* 24, 3 (July 2005), 965–972. DOI: http://dx.doi.org/10.1145/1073204.1073298