Unified Particle Physics for Real-Time Applications

Miles Macklin* Matthias Müller† Nuttapong Chentanez‡ Tae-Yong Kim§
NVIDIA

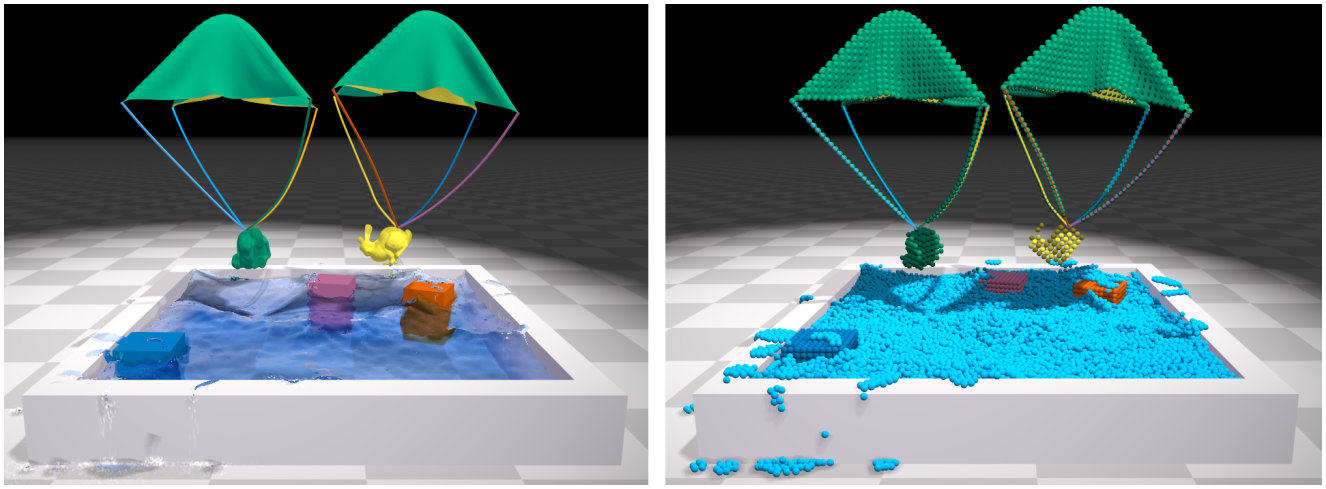

Figure 1: *Bunnies parachute into a pool of water. Cloth, rigid bodies and fluids coupled through constraints interact seamlessly in our framework.*

Abstract

We present a unified dynamics framework for real-time visual effects. Using particles connected by constraints as our fundamental building block allows us to treat contact and collisions in a unified manner, and we show how this representation is flexible enough to model gases, liquids, deformable solids, rigid bodies and cloth with two-way interactions. We address some common problems with traditional particle-based methods and describe a parallel constraint solver based on position-based dynamics that is efficient enough for real-time applications.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: simulation, fluid simulation, unified solver, position-based dynamics, two-way fluid coupling, cloth simulation

Links: DL PDF WEB VIDEO

*e-mail:mmacklin@nvidia.com
†e-mail:matthiasm@nvidia.com
‡e-mail:nchentanez@nvidia.com
§e-mail:taeyongk@nvidia.com

ACM Reference Format
Macklin, M., Möller, M., Chentanez, N., Kim, T. 2014. Unified Particle Physics for Real-Time Applications. ACM Trans. Graph. 33, 4, Article 153 (July 2014), 12 pages. DOI = 10.1145/2601097.2601152 http://doi.acm.org/10.1145/2601097.2601152.

0730-0301/14/07-ART153 $15.00.
DOI: http://dx.doi.org/10.1145/2601097.2601152

1 Introduction

Unified solvers have become popular tools in offline visual effects in recent years. Notable examples include Maya's Nucleus solver [Stam 2009] and Softimage's Lagoa. We were inspired by the features of these packages, and our goal here is to create a unified solver capable of simulating gases, liquids, deformable solids, rigid bodies, and cloth, interacting with one another in real-time.

The concept of a unified dynamics solver is attractive. Having a single piece of software that can simulate multiple object types that were previously handled by specialized solvers means less code to write, optimize, and maintain. Additionally, objects that were previously simulated in isolation can now interact with each other in a fully coupled way. In this paper we present a unified approach that makes some compromises in realism to meet our goal of real-time performance, but enables a wider range of effects than was previously possible. Our framework is built on the position-based dynamics method (PBD) [Müller et al. 2007] which is popular in computer games and interactive applications for its simplicity and stability.

Our paper is organized as follows: first we discuss our particle-based representation (section 3), we then present our general purpose parallel constraint solver (section 4) followed by the constraint types used in our solver (sections 5-8). Our main contributions are summarized below:

- Extending constraint averaging to PBD to support parallel execution (section 4)
- Approximate shock propagation to increase convergence of rigid stacks (section 5.2)
- Static and dynamic friction in PBD (section 6.1)
- Two-way fluid solid coupling (section 7.1)
- Gas simulation with PBD (section 7.2)

2 Related Work

There has been increasing work towards unified simulation models recently, and point-based methods are well suited to the problem. Müller et al. [2004a] use a point based representation to model elastic and plastic solids that can topologically deform. Solenthaler et al. [2007] described a method for simulating fluids and solids based on smoothed particle hydrodynamics (SPH). Becker et al. [2009] improve the rotational invariance of their method using a co-rotated deformation model based on SPH. Martin et al. [2010] address issues with degenerate particle configurations using $elastons$ which provide a more accurate measure of local deformation. In contrast to these methods we do not use a unified constitutive model based on continuum mechanics. Rather, we prefer to combine custom position-level constraints for each material we wish to simulate.

Gascuel and Gascuel [1994] used displacement constraints to animate rigid bodies and articulated figures. Their work was extended by Faure [1999], and position-based dynamics [Müller et al. 2007] can be seen as a generalization of their method. To simulate rigid bodies, Gascuel and Gascuel [1994] converted position displacements into rigid transformations by first finding a suitable rotation and then applying a matching translation. Their method favors rotation over translation, so we instead use rigid shape-matching [Müller et al. 2005] which finds the least-squares best transform. Stam [2009] built a unified solver using constrained simplices, while in the context of real-time simulation, Jakobsen [2001] used particles connected by constraints to model articulated characters. Müller and Chentanez [2011b] use an oriented particle representation connected by shape matching and distance constraints to model elastic and plastic solids.

One aspect of unified simulation of particular interest is two-way interaction of fluids and solids. Carlson et al. [2004] presented a method to couple rigid bodies with a grid-based fluid simulation. Carlson [2004] also simulate rigid, melting and flowing materials in a unified way. Guendelman et al. [2005] show how to couple Eulerian fluid simulations to solid and thin-shell deformables. A full discussion of grid-based methods is, however, beyond the scope of our paper. Coupling particle fluids with rigid bodies was explored by Müller et al. [2004b] who couple a SPH fluid simulation to mesh based deformable objects. Clavet et al. [2005] couple fluids with rigid bodies by treating particles as hard spheres and applying impulses to a traditional rigid body simulator. Akinci et al. [2012] couple SPH fluids with rigid bodies using more accurate boundary force calculations, and extend their method to handle deformable solids in [Akinci et al. 2013b]. In contrast to these works, we simulate fluids and rigid bodies inside the same framework.

In addition to the work discussed here, we will refer to more specific related work in the respective sections.

3 Particle Representation

We choose particles as the fundamental building block for all object types. Particles are attractive for their simplicity and ease of implementation, while being flexible enough to represent the range of objects we wish to simulate. By constructing all objects from particles, we significantly reduce the number of collision types we need to process, and avoid complex algorithms for generating contacts between mesh based representations. Particles also provide a natural granularity at which to express parallelism. To take advantage of modern graphics processors (GPUs) we prefer algorithms that consider many simple particle interactions in parallel, over more advanced algorithms that run serially.

Our core particle state consists of the following properties:

```
struct Particle
{
    float x[3];
    float v[3];
    float invmass;
    int phase;
};
```

We extend the common dynamics quantities with a particle phase identifier. This integral value is used to organize particles into groups, and provides a convenient way to adjust their properties and control how they interact, for example, disabling collisions between groups of particles. We restrict ourselves to a fixed particle radius per scene in order to leverage efficient collision detection based on uniform grids.

4 Parallel SOR Solver

4.1 Background

Our system is built around a general purpose constraint solver that is derived from position-based dynamics. We give some background on position-based dynamics here, but refer the reader to [Müller et al. 2007] and [Bender et al. 2014] for more information. PBD solves a system of non-linear equality and inequality constraints such that

$$C_i(\mathbf{x}+\mathbf{\Delta x}) = 0, \quad i = 1,\dots,n \tag{1}$$
$$C_j(\mathbf{x}+\mathbf{\Delta x}) \geq 0, \quad j = 1,\dots,n. \tag{2}$$

where $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T$ is the vector of particle positions. Constraints are typically solved through Gauss-Seidel iteration, where each constraint is solved sequentially using the linearization of C around $\mathbf{x}$,

$$C_i(\mathbf{x}+\mathbf{\Delta x}) \approx C_i(\mathbf{x}) + \nabla C_i(\mathbf{x})\mathbf{\Delta x} = 0. \tag{3}$$

The position change $\mathbf{\Delta x}$, is restricted to lie along the constraint gradient, and is weighted by the inverse of the mass matrix $\mathbf{M} = diag(m_1, \cdots, m_n)$,

$$\mathbf{\Delta x} = \mathbf{M}^{-1}\nabla C_i(\mathbf{x})^T\lambda_i. \tag{4}$$

Combining Eq. (3) and (4), λ_i is given by

$$\lambda_i = -\frac{C_i(\mathbf{x})}{\nabla C_i(\mathbf{x})\mathbf{M}^{-1}\nabla C_i(\mathbf{x})^T}. \tag{5}$$

Typically positions are updated after each constraint is processed, and after a number of iterations a change in velocity is computed according to the total constraint delta

$$\mathbf{\Delta v} = \frac{\mathbf{\Delta x}}{\Delta t}. \tag{6}$$

4.2 Optimization Viewpoint

Here we present another perspective on position-based dynamics by viewing it as the solution to a constrained optimization problem. Considering only equality constraints, the problem to be solved can be stated as:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\mathbf{\Delta x}^T\mathbf{M}\mathbf{\Delta x} \\ &\text{subject to} && C_i(\mathbf{x}+\mathbf{\Delta x}) = 0, \quad i = 1,\dots,n \end{aligned} \tag{7}$$

The solution variable $\mathbf{\Delta x}$ is the change in position such that the constraints are satisfied. According to Eq. (6) the resulting change in position is equivalent to applying an impulse at the beginning of the time-step. As such, the problem is equivalent to finding the minimum change in kinetic energy that satisfies the constraints, which is consistent with Gauss's principle of least constraint.

If the constraint functions were linear, then (7) would be a constrained quadratic minimization problem (QP) with a closed form solution. In practice, the constraints are arbitrary non-linear, non-convex functions, for which fast and globally optimal solvers do not exist [Boyd and Vandenberghe 2004]. Position-based dynamics proceeds in the spirit of sequential quadratic programming (SQP) by linearizing the constraints and solving a sequence of local constrained quadratic minimizations:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\mathbf{\Delta x}^T\mathbf{M}\mathbf{\Delta x} \\ &\text{subject to} && \mathbf{J}\mathbf{\Delta x} = \mathbf{b} \end{aligned} \tag{8}$$

where $\mathbf{J} = \nabla C(x)$, and $\mathbf{b} = [-C_i, \ldots, -C_n]^T$. Problems of this type (minimizing a quadratic objective with linear constraints) can be transformed into the following optimality conditions:

$$\mathbf{M}\mathbf{\Delta x} = \nabla\mathbf{J}^T\boldsymbol{\lambda} \tag{9}$$

$$\mathbf{J}\mathbf{\Delta x} = \mathbf{b} \tag{10}$$

The first condition comes from the theory of Lagrange multipliers where the left-hand side is the gradient of the objective function, and the second condition comes from from the feasibility requirement. Equations (4) and (5) follow immediately from these conditions, and when considering the system as a whole, we can eliminate $\mathbf{\Delta x}$ and write

$$\left[\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right]\boldsymbol{\lambda} = \mathbf{b}. \tag{11}$$

Eq. (11) is a matrix equation for the linearized problem. This is similar to the fast projection of Goldenthal et al. [2007]. However, in PBD, the matrix is never explicitly formed, and rather than solve for $\boldsymbol{\lambda}$ exactly, PBD applies a single Gauss-Seidel iteration over (11) before updating $\mathbf{x}$ and restarting the problem. Projected Gauss-Seidel is used in the presence of inequality constraints.

Gauss-Seidel iteration is inherently serial, so to increase parallelism we solve constraints in a projected Gauss-Jacobi fashion. Unfortunately, Jacobi iteration is not guaranteed to converge if the system matrix is not positive definite. In some cases this is not a problem and Macklin and Müller [2013] successfully used Jacobi iteration to simulate fluids in the position-based framework using constraint regularization. However, for many common constraint configurations it is clear that a solution will not be reached. We illustrate the problem by considering a 1-dimensional particle constrained to the origin by two identical distance constraints (Figure 2).

Figure 2: *A particle constrained to lie at the origin by two identical distance constraints.*

In this example, the system of constraint equations to solve is:

$$C_1(x) = x = 0$$
$$C_2(x) = x = 0$$

Because the two constraints are identical, $\mathbf{J}$ is rank deficient, and the system matrix is singular. Although Gauss-Seidel iteration would find a solution, Jacobi iteration will oscillate between two fixed solutions indefinitely (the positive and negative side of the origin). If the constraints are not identical but merely close, then the condition number of the matrix is large and convergence will be slow. Situations like this are common in practice.

To address this problem, we apply under-relaxation based on the concept of constraint averaging [Bridson et al. 2002], or mass-splitting [Tonge et al. 2012]. We first process each constraint in parallel and accumulate position deltas for each particle. At the end of the iteration, once all constraints are processed, we divide each particle's total constraint delta by the number of constraints affecting it,

$$\mathbf{\Delta x_i} = \frac{1}{n}\sum_n \nabla C_j \lambda_j. \tag{12}$$

This form of local relaxation is not guaranteed to conserve momentum when neighboring particles have differing number of constraints, however visual errors are typically not noticeable.

4.3 Successive Over-Relaxation

Averaging constraint forces as described above ensures convergence, but in some cases this averaging is too aggressive and the number of iterations required to reach a solution increases. To address this we introduce a global user-parameter ω which controls the rate of successive over-relaxation (SOR),

$$\mathbf{\Delta x}_i = \frac{\omega}{n}\sum_n \nabla C_j \lambda_j. \tag{13}$$

In all our simulations we have used $1 \leq \omega \leq 2$, although higher values may be used depending on the scene being simulated. Additional under-relaxation ($\omega < 1$) is not typically required as the constraint averaging is sufficient to avoid divergence.

In practice it can be desirable to have some constraint types take precedence over others. To achieve this, we process constraints in groups and immediately apply the accumulated position delta to the particle positions before the next constraint group is processed. Typically, all constraints of the same type are processed in the same group. For example, after processing all density constraints in parallel, the accumulated delta can be applied to our candidate position $\mathbf{x}^*$ before processing contact constraints in parallel (steps 17-21 in Algorithm 1). This method also propagates constraint corrections faster which improves convergence speed.

4.4 Initial Conditions

A common situation is for constraints to be violated at the beginning of a time-step. This could occur, for example, due to convergence not being reached at the end of the last step, or a kinematic object being moved to an invalid state in response to user input. Due to the way position-based dynamics solves constraints, these invalid initial conditions may cause kinetic energy to be added to the system in an undesirable manner. To illustrate the problem, we consider a particle initially interpenetrating the ground (Figure 4). The position-based solver calculates the corrected position at the surface $\mathbf{x}^*$, projects the particle there and updates the velocity according Eq. (6).

This new velocity causes the particle to continue to travel upwards and 'pop' out of the ground. What's worse, this incorrect velocity becomes larger as the time-step decreases! This is a serious problem for stable stacking and can cause obvious non-physical

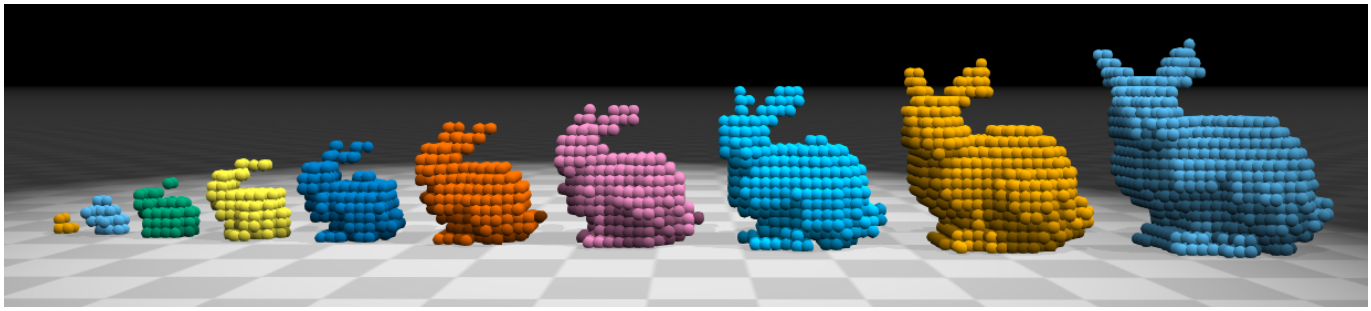

Figure 3: *Showing the Stanford bunny at different sizes sampled by particles.*

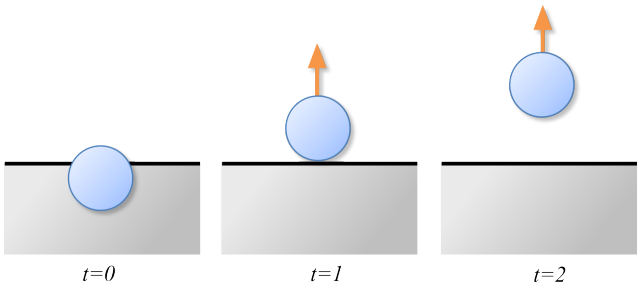

Figure 4: *A particle with zero velocity is interpenetrating the ground at the beginning of the time-step (left). Position-based dynamics projects the particle to the surface and updates velocity according to this movement (middle). The resulting velocity causes a change of momentum and the particle to continue out of the ground (right).*

behavior such as cloth bouncing off the ground. One way to view this correction of initial error is as a Baumgarte stabilization [Ascher et al. 1995] with a stiffness factor of 1. The differential algebraic equations (DAE) community have long known about the drawbacks of Baumgarte stabilization and have moved to pre- and post-stabilization techniques that modify position independent of velocity. For further reading we refer to [Weinstein et al. 2006], which provides a good overview of stabilization methods from a computer graphics perspective.

We address this issue by optionally performing a pre-stabilization pass on the initial positions in order to move particles to a valid state. The process involves first predicting new positions, generating contacts, and then solving the resulting contact constraints with the original, rather than predicted positions. Any deltas applied to the original positions are also applied to the predicted positions before the main constraint solving loop begins. In practice we perform this pre-stabilization only for contacts, as this is the most visible source of error. If convergence is not fully reached during this pre-stabilization pass then some energy may still be added to the system, however 1-2 iterations is usually sufficient to resolve most visual artifacts.

4.5 Particle Sleeping

Positional drift may occur when constraints are not fully satisfied at the end of a time-step. We address this by freezing particles in place if their velocity has dropped below a user-defined threshold,

$$\mathbf{x}(\mathbf{t}+\mathbf{\Delta t}) = \begin{cases} \mathbf{x}^*, & |\mathbf{x}^* - \mathbf{x}_0| > \epsilon \\ \mathbf{x}_0, & otherwise \end{cases} \tag{14}$$

Algorithm 1 Simulation Loop

for all particles i **do**
  apply forces $\mathbf{v}_i \Leftarrow \mathbf{v}_i + \Delta t \mathbf{f}_{ext}(\mathbf{x}_i)$
  predict position $\mathbf{x}_i^* \Leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
  apply mass scaling $m_i^* = m_i e^{-kh(\mathbf{x}_i^*)}$
end for
for all particles i **do**
  find neighboring particles $N_i(\mathbf{x}_i^*)$
  find solid contacts
end for
while $iter < stabilizationIterations$ **do**
  $\Delta\mathbf{x} \Leftarrow \mathbf{0}, n \Leftarrow 0$
  solve contact constraints for $\Delta\mathbf{x}, n$
  update $\mathbf{x}_i \Leftarrow \mathbf{x}_i + \Delta\mathbf{x}/n$
  update $\mathbf{x}^* \Leftarrow \mathbf{x}^* + \Delta\mathbf{x}/n$
end while
while $iter < solverIterations$ **do**
  for each constraint group G **do**
    $\Delta\mathbf{x} \Leftarrow \mathbf{0}, n \Leftarrow 0$
    solve all constraints in G for $\Delta\mathbf{x}, n$
    update $\mathbf{x}^* \Leftarrow \mathbf{x}^* + \Delta\mathbf{x}/n$
  end for
end while
for all particles i **do**
  update velocity $\mathbf{v}_i \Leftarrow \frac{1}{\Delta t}(\mathbf{x}_i^* - \mathbf{x}_i)$
  advect diffuse particles
  apply internal forces $\mathbf{f}_{drag}, \mathbf{f}_{vort}$
  update positions $\mathbf{x}_i \Leftarrow \mathbf{x}_i^*$ or apply sleeping
end for

5 Rigid Bodies

Harada [2007] and Bell et al. [2005] used a particle representation to simulate rigid bodies with a penalty force model, while Tonge et al. [2010] used particles to detect contacts for a constraint based GPU rigid body solver. An alternative approach is to construct rigid bodies by connecting particles with a lattice of distance constraints, however this method requires many iterations to appear stiff.

We represent non-convex rigid bodies using particles, and use rigid-shape matching constraints [Müller et al. 2005] to maintain particle configurations. Shapes are created by first performing solid voxelization of a closed triangle mesh and placing particles at occupied cells interior to the shape. We then assign all particles in the shape the same phase-identifier, disable collision between them, and add a shape-matching constraint to the system.

During simulation we treat particles as if they were unconnected, subjecting them to displacements from the other constraint types. We then find the least squares best transform that maps the rest-

positions to the deformed state and transform particles accordingly (Figure 5). This deformation and re-targeting is similar to the approach taken by Bao et al. [2007] in the context of a finite element simulation.

The position delta due to a shape-matching constraint is given by

$$\mathbf{\Delta x}_i = (\mathbf{Q r}_i + \mathbf{c}) - \mathbf{x}_i^* \tag{15}$$

where $\mathbf{Q}$ is a rotation matrix given by the polar-decomposition of the deformed shape's covariance matrix $\mathbf{A}$, calculated as:

$$\mathbf{A} = \sum_i^n (\mathbf{x}_i^* - \mathbf{c}) \cdot \mathbf{r}_i^T \tag{16}$$

where $\mathbf{r}_i$ is the particle's offset from the center of the mass in the rest configuration, and $\mathbf{c}$ is the center of mass of the particles in the deformed configuration. The evaluation of Eq. (16) can be performed efficiently in parallel by assigning a thread per particle to calculate each outer product and then forming $\mathbf{A}$ using a parallel reduction across all threads in the shape.

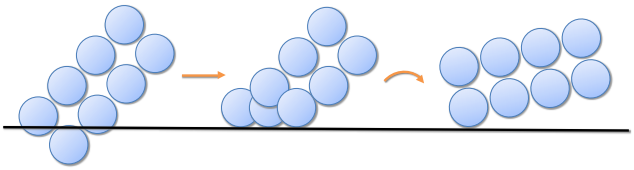

Figure 5: *Showing how shape matching constraints map deformed particles back to a rigid pose.*

We note that using this method we never explicitly store the rigid body's linear or angular velocity, as these are given implicitly by the particle state. The shape matching step automatically respects the inertial properties of the object, making this a very simple rigid body simulator.

One limitation of this method is that, when using particles with a fixed radius r, the number of particles required to represent a given shape grows proportional to $O(n^3)$ where $n = \frac{1}{r}$. This places a limit on the maximum size ratio that can be efficiently represented, so for real-time simulations we suggest a rule of thumb of 1:10 between the smallest and largest feature (see Figure 3). Placing particles in a thin layer around the surface can reduce the number of particles required to $O(n^2)$, although this places some time-step restrictions in order to prevent tunneling.

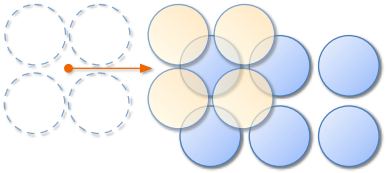

Figure 6: *Rigid groups of particles can interpenetrate and become locked together due to discrete collision handling.*

5.1 Sparse Signed Distance Field Collision

To resolve collisions, Harada [2007] used discrete element forces between pairs of particles belonging to rigid bodies. The simplicity of this approach is attractive as only local collisions between particle pairs need be considered. However, if tunneling occurs then bodies can interlock and fail to separate (Figure 6). We address this problem using a new particle collision algorithm based on a sparse signed distance field (SDF) representation.

Guendelman et al. [2003] used signed distance fields to generate contacts between non-convex rigid bodies by point sampling triangle mesh features within each overlapping shape's SDF. Collisions are then resolved using an sequential, iterative scheme where each feature is projected out of the shape in order of deepest penetration first. In contrast to Guendelman et al. [2003] who use grid-based SDFs, we sample our field function onto each particle belonging to a rigid shape. By storing the SDF on particles we are able to re-use all the machinery of our particle-particle collision detection to resolve deeper overlaps between shapes. We store both the magnitude ϕ and gradient $\nabla\phi$ of the SDF at each particle's location (Figure 7) which can be viewed as an approximate first order, sparse representation of the underlying field. Higher order representations are also possible, as described in [Corbett 2005].

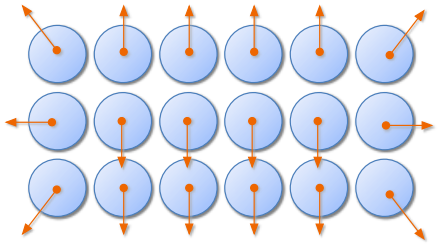

Figure 7: *We store a signed distance field value and its gradient per particle to prevent rigid interpenetration and locking. Note that particles on the medial axis are assigned a gradient direction arbitrarily.*

To resolve collisions with this representation we first detect overlaps between particles with distance $|\mathbf{x}_i - \mathbf{x}_j| < r$. As the contact normal we choose the SDF gradient corresponding to the minimum translation distance, $d = \min(|\phi_i|, |\phi_j|)$.

$$\mathbf{n}_{ij} = \begin{cases} \nabla\phi_i & \text{if } |\phi_i| < |\phi_j| \\ -\nabla\phi_j & \text{otherwise} \end{cases} \tag{17}$$

The position delta for each particle is then given by

$$\mathbf{\Delta x}_i = -\frac{w_i}{w_i + w_j}(d \cdot \mathbf{n}_{ij}) \tag{18}$$

$$\mathbf{\Delta x}_j = \frac{w_j}{w_i + w_j}(d \cdot \mathbf{n}_{ij}), \tag{19}$$

where $w_i = 1/m_i$.

The signed distance field is often undersampled near the surface of a shape which can lead to discontinuous penetration depths and jittering. To address this we treat boundary particles (those with $|\phi| < r$) separately from the method for interior particles described above. When a collision between a boundary particle and another particle is detected, we modify the contact normal according to [Müller and Chentanez 2011a], and use $d = |\mathbf{x}_i - \mathbf{x}_j| - r$, effectively treating boundary particles as one-sided hard spheres.

This modified boundary normal (Figure 8) can be computed efficiently as the reflection of the collision direction around the SDF gradient (from Eq. 17) when a particle lies in the negative half-space of the particle.

$$\mathbf{n}_{ij}^* = \begin{cases} \mathbf{x}_{ij} - 2(\mathbf{x}_{ij} \cdot \mathbf{n}_{ij})\mathbf{n}_{ij} & \mathbf{x}_{ij} \cdot \mathbf{n}_{ij} < 0 \\ \mathbf{x}_{ij} & \text{otherwise} \end{cases} \tag{20}$$

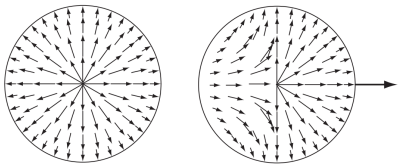

Figure 8: *Left: regular particle collision normals. Right: one sided collision normals used for particles on the rigid shape's boundary.*

A limitation of this approach is that particle surfaces are not entirely smooth. To reduce bumping and artificial sticking, we ensure some overlap between particles in the rest pose.

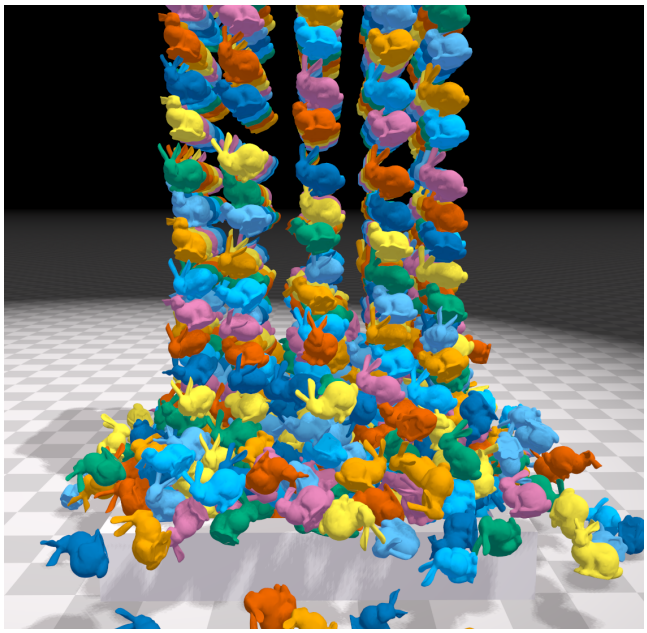

Figure 9: *1000 non-convex objects, each consisting of 44 particles, form a pile. Two-substeps and two-constraint iterations are computed in 4ms/frame.*

5.2 Stiff Stacks

Jacobi methods can only propagate information (collision deltas) over a distance of one particle per iteration. This can lead to large of piles rigid bodies requiring many iterations to appear stiff. One method for increasing the rate of information propagation is shock propagation [Guendelman et al. 2003]. Shock propagation works by iterating bodies from the ground upwards, fixing each layer of bodies in place after they have been processed. Unfortunately this process is inherently serial (layers are processed one at a time in a sequential fashion). Here we present a new, parallel-friendly method to improve the stability of rigid stacks that does not require changing iteration order.

First, we estimate the stack height of each particle, h. This may be done through a contact graph search, or field propagation, from the solid boundary. The main requirement is that the height function increases in the direction opposite gravity. In our examples we have used a simple heuristic function measuring height from a fixed ground plane.

Once we have an estimate of stack height, we temporarily modify each particle's mass such that lower particles have a larger mass than the ones above. In our experiments, we found the following exponential function works well as a scale factor for particle mass,

$$s_i(\mathbf{x}_i) = e^{-kh(\mathbf{x}_i)} \tag{21}$$

where $h(\mathbf{x}_i)$ is the stack height for a particle. In the case that $h(\mathbf{x_i})$ is equal to height from the ground plane then this scaling function provides a constant mass ratio of $s_i/s_j = e^{-kr}$ for two particles i and j stacked a distance r apart. The resulting mass ratio causes lower particles to feel less pressure and to resist compression. This causes stacks of bodies to converge to a solution much faster (Figure 10). In our rigid piling scenes we have typically used values of $k \in [1 \ldots 5]$.

The scaled particle mass $m_i^* = s_i m_i$ is used only during the contact processing phase and is reset post-constraint solve. Because each particle is processed independently, we can perform this mass adjustment in parallel (step 4 in Algorithm 1).

We note that our method is not specific to particle-based solvers and may be applied equally well in the context of a traditional rigid body solver. However a potential limitation of this method is that if the coefficient k is set too high then the lower particles will stop responding to interaction from above. One solution to this problem is to perform mass modification only in the final solver iteration, however we have not found this to be necessary in order to achieve good results.

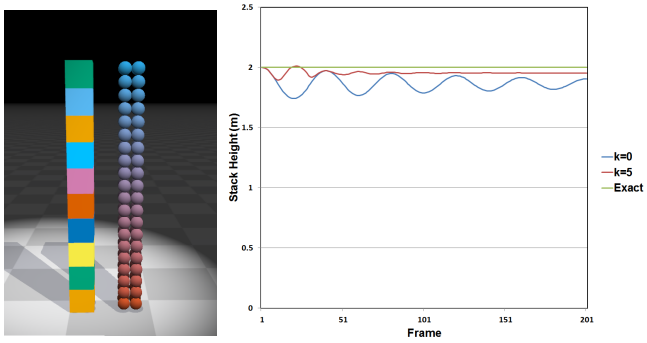

Figure 10: *Left: A 1x1x10 stack of rigid bodies 2m high coming to rest under gravity. Particles are color coded by our stack height function (section 5.2). Right: without our mass modification the stack oscillates for a long time and shows significant compression (blue). Using our method (red) the stack stabilizes quickly and comes to rest closer to the true solution (green).*

5.3 Deformation

Each shape-matching constraint has an associated stiffness parameter that allows us to model small scale elastic deformation. In addition, we can deform objects plastically using the method described in [Müller and Chentanez 2011b]. Briefly, we detect when a particle belonging to a rigid shape has been deformed past a user threshold. We then mix this deformation back into the particle's local-space rest position according to a plastic creep coefficient. Because this process invalidates our SDF field, we convert particles back to regular spherical particles if they deform past a certain threshold.

Currently we use a single shape-matching constraint per object, which limits the amount of deformation possible. Larger deformations can be supported by combining multiple shape-matching constraints, or using methods such as oriented particles or tetrahedral volume constraints, both of which can be accommodated in our constraint framework.

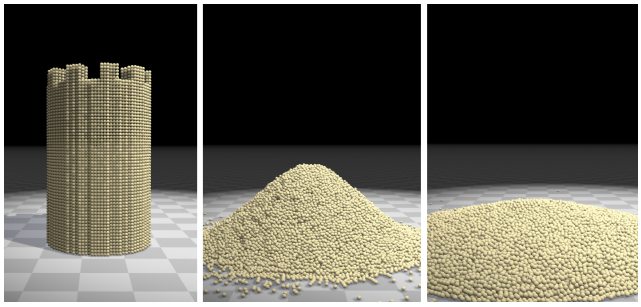

Figure 11: *A sand castle before collapse (left). After 300 frames our position-based friction model maintains a steep pile (middle), while the original position-based dynamics friction model has almost completely collapsed (right).*

6 Granular Materials and Friction

We present our friction model in the context of granular materials whose behavior is heavily dependent on frictional effects. We note, however, that the same method presented here is used for all object types in our framework.

Previous work by Bell et al. [2005] modeled granular materials using the discrete element method and irregular groups of particles. Zhu and Bridson [2005] used the FLIP method to animate sand, while Alduán and Otaduy [2011], and Ihmsen et al. [2012b] used iterative Smoothed Particle Hydrodynamics (SPH) solvers to model granular materials with friction and cohesion. Position-based dynamics traditionally models friction by damping velocity after the position-based constraint solve has completed [Müller et al. 2007]. This method cannot model static friction because the frictional damping forces cannot correct for the position changes already made during the constraint solve. Consequently, particle positions drift, and piles quickly collapse. We address this problem using a novel formulation that applies friction at the position level.

6.1 Friction Model

During contact handling, we first resolve interpenetration by projecting particles a distance d along the collision normal according to the following non-penetration constraint,

$$C(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_{ij}| - r \geq 0. \tag{22}$$

Once interpenetration has been resolved, we calculate a frictional position delta based on the relative tangential displacement of the particles during this time-step

$$\mathbf{\Delta x}_\perp = \left[(\mathbf{x}_i^* - \mathbf{x}_i) - (\mathbf{x}_j^* - \mathbf{x}_j)\right] \perp \mathbf{n}, \tag{23}$$

where $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ are the current candidate positions for the colliding particles including any previously applied constraint deltas, $\mathbf{x}_i$ and $\mathbf{x}_j$ are the positions of the particles at the start of the time-step, and $\mathbf{n} = \mathbf{x}_{ij}^*/|\mathbf{x}_{ij}^*|$ is the contact normal. The frictional position delta for particle i is then computed as

$$\mathbf{\Delta x}_i = \frac{w_i}{w_i + w_j} \begin{cases} \mathbf{\Delta x}_\perp, & |\mathbf{\Delta x}_\perp| < \mu_s d \\ \mathbf{\Delta x}_\perp \cdot \min(\frac{\mu_k d}{|\mathbf{\Delta x}_\perp|}, 1), & otherwise \end{cases} \tag{24}$$

where μ_k, μ_s are the coefficients of kinetic and static friction respectively. The first case in Eq. (24) models static friction by removing all tangential movement when the particle's relative velocity is below the traction threshold. The second case models kinetic Coulomb friction, limiting the frictional position delta based on the penetration depth of the particle. The position change on particle j is given by

$$\mathbf{\Delta x}_j = -\frac{w_j}{w_i + w_j} \mathbf{\Delta x}_i. \tag{25}$$

Treating the frictional impulses this way allows us to generate large particle piles with high angles of repose (Figure 11). However, as with position-based dynamics generally, a limitation is that friction strength is somewhat dependent on iteration count. Müller et al. [2007] suggest a method to reduce this effect.

7 Fluids

We simulate fluids in our framework using the position-based fluids method [Macklin and Müller 2013], where the fluid density constraint (Eq. 26) is considered simply as another constraint in the system. In contrast to [Macklin and Müller 2013] we clamp the density constraint to be non-negative (unilateral), so that it only acts to separate particles,

$$C(\mathbf{x}_1, ..., \mathbf{x}_n) = \frac{\rho_i}{\rho_0} - 1 \leq 0, \tag{26}$$

we then use the model of [Akinci et al. 2013a] to handle cohesion and surface tension effects.

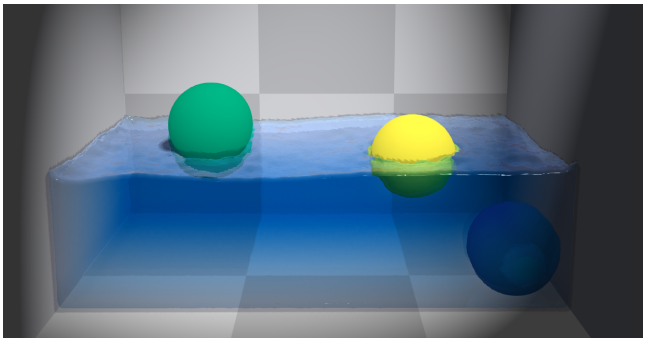

Figure 12: *Assigning particles different masses gives rise to buoyancy.*

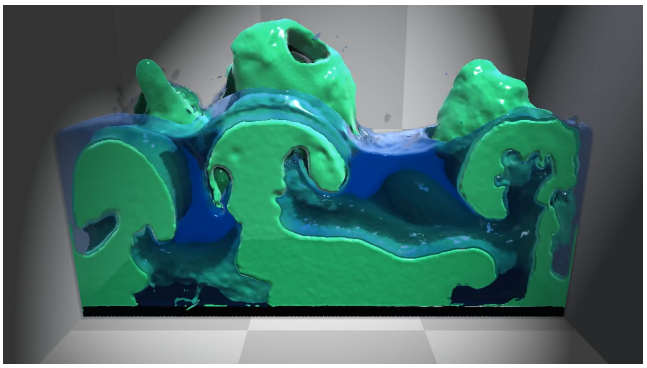

Figure 13: *Two-phase liquid with a density ratio of 4:1 showing the Rayleigh-Taylor instability.*

7.1 Fluid-Solid Coupling

Akinci et al. [2012] show how to accurately couple SPH fluids with rigid bodies by placing particles at solid boundaries and calculating

fluid pressure forces which are then input to a traditional rigid body simulator. Our time-steps are typically large enough that we cannot rely on fluid pressure to prevent interpenetration and tunneling. Consequently, when a fluid particle comes into contact with a solid particle, we treat both as solid particles with the fluid rest distance used as the contact distance r.

We include solid particles in the fluid density estimation, so the density for a fluid particle is then given by

$$\rho_i = \sum_{fluid} W(\mathbf{x}_i - \mathbf{x}_j, h) + s \sum_{solid} W(\mathbf{x}_i - \mathbf{x}_j, h) \qquad (27)$$

where h is width of the smoothing kernel W. The parameter s accounts for any difference in sampling density between solid particles and fluid particles. If the distance between fluid particles at the rest density is the same as solid particles then s can be set to 1. If solids are sampled more densely than fluids then it should be set < 1. Because s is a constant, we assume solid particles are sampled at relatively uniform density. Akinci et al. [2012] use a per-particle correction, however we have not found this to be necessary, partly because we typically use regular particle samplings, and also because we do not rely on fluid pressure forces to resolve fluid-solid interactions.

7.1.1 Buoyancy

The density constraint formulation of [Macklin and Müller 2013] treats fluid particles as having equal mass, but by using the mass weighted version of position-based dynamics (Eq. 4) we can simulate fluids and rigid bodies with differing densities. This simple adjustment automatically gives rise to buoyancy and sinking of objects with differing mass ratios (Figure 12).

For the fluid density constraints, we pre-compute the denominator in Eq. (5) for a reference filled particle configuration. During this pre-computation step we assume that neighboring particles have the same mass, so to ensure the subsequent position corrections from Eq. (4) are conservative, the smallest expected particle mass in the system should be used.

To simulate immiscible multi-phase liquids (Figure 13) we allow fluid particles of different phases to contribute to each other's density estimation. We apply cohesive and surface tension forces only between particles of the same phase as in [Solenthaler and Pajarola 2008].

7.2 Gases

Grid-based methods have proved popular for animating gases and smoke, in particular [Stam 1999] and [Fedkiw et al. 2001]. More recently, Lagrangian discretizations for smoke have gained interest in computer graphics, notably vortex methods such as [Park and Kim 2005], and mesh-tracking methods [Pfaff et al. 2012][Brochu et al. 2012]. Stam and Fiume [1995] were the first to apply SPH to the simulation of gas and fire, and Gao et al. [2009] combined SPH with a grid-based fluid solver to simulate fast moving gases.

We present a fully Lagrangian method for simulating gases based on position-based fluids. Our method is sparse in the sense that it only computes fluid dynamics in areas of visual interest, and, in contrast to vortex particle methods which require special handling for solid boundaries, our method handles solid boundary conditions through simple position projection during the regular particle collision constraint pipeline.

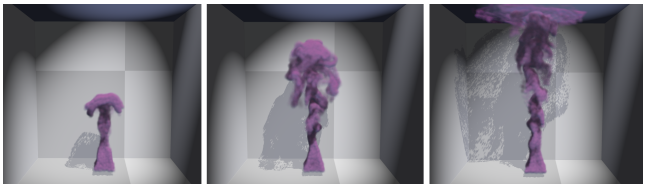

Figure 14: *A rising smoke plume simulated using position-based fluids, this example uses 16k fluid particles (32x32x16) and 65k smoke particles.*

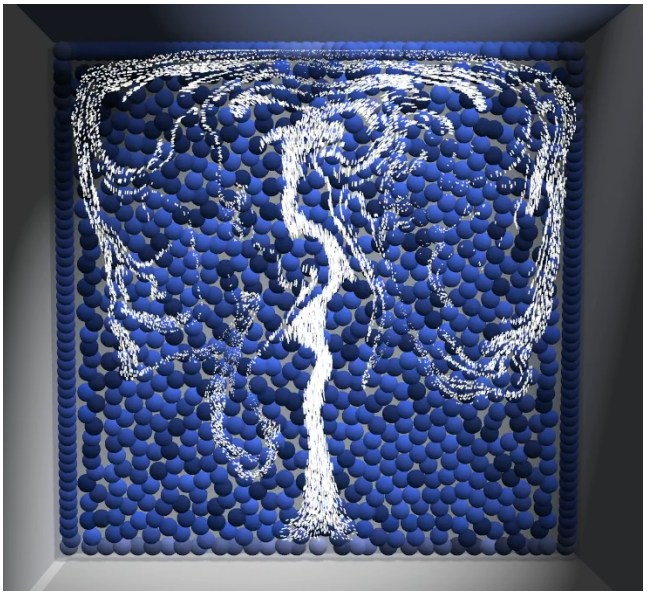

Figure 15: *Slice from a smoke simulation with closed boundaries. The domain is seeded with fluid particles (blue) and smoke particles (white) are passively advected through the velocity field.*

7.2.1 Closed Boundaries

To model smoke in closed environments, we begin by filling the computation domain with fluid particles and reducing gravity for gas particles according to a user parameter α, although a temperature-based buoyancy model would work equally well if tracking temperature per particle. We treat the gas as incompressible and use the unilateral density constraint to maintain a fixed rest density.

We inject visual smoke particles into the simulation domain and passively advect them through the velocity field defined on the fluid particles (Figure 14). To calculate the velocity at a smoke particle's location $\mathbf{x}_s$ we use the following weighted averaging over fluid particles:

$$\mathbf{v}(\mathbf{x}_s) = \frac{\sum_j \mathbf{v}_j W(\mathbf{x}_s - \mathbf{x}_j)}{\sum_j W(\mathbf{x}_s - \mathbf{x}_j)} \qquad (28)$$

where $W(\mathbf{x})$ is any symmetric kernel function, e.g. the cubic Poly6 kernel given in [Müller et al. 2003], and $\mathbf{v}_j$ is the velocity of a fluid particle which overlaps the sampling location. Note that we use the velocity calculated immediately after the constraint solve to ensure the velocity used for advection is consistent with the incompressibility constraints (step 24 in Algorithm 1). We note that these passive visual marker particles are useful in other contexts, for example we also use them to represent spray and foam particles in liquid simulation as in [Ihmsen et al. 2012a].

7.2.2 Open Boundaries

To generate plumes of smoke, we inject fluid particles and smoke particles together, ensuring a 1-2 layer thick boundary of fluid particles surrounding the smoke particles at their emission point. Interior fluid particles are emitted with a higher velocity than the boundary particles, and to model the effect of fast moving air interacting with the surrounding environment we introduce a new drag force:

$$\mathbf{f}_{drag_i} = -k(\mathbf{v}_i - \mathbf{v}_{env})(1 - \frac{\rho_i}{\rho_0}) \tag{29}$$

where $\mathbf{v}_{env}$ is the environmental velocity at the particle's location. In our examples, $\mathbf{v_{env}}$ has been taken to be $\mathbf{0}$ to model still air, however any procedural velocity field may be used. The last factor on the right-hand side of Eq. (29) ensures that only particles near the free surface experience drag. If smoke particles become isolated from the fluid we revert them to ballistic motion, or advect them by the procedural background velocity field. Additionally, fluid particles may be removed once they have no smoke particles within their kernel radius, or after a pre-defined lifetime.

Because we do not simulate fluid where smoke does not exist, this technique is not well suited to simulating long range effects, e.g.: a gust of wind from a window disturbing smoke at the other end of the room. However these types of effects may be approximated using procedural techniques that modify the environment velocity in Eq. (29). It is also important to ensure that there is a sufficient sampling of fluid particles to provide accurate velocity estimates for smoke particle advection. We use weak cohesive forces to maintain particle density, although dynamic particle seeding strategies such as Ghost SPH [Schechter and Bridson 2012] may be a better alternative.

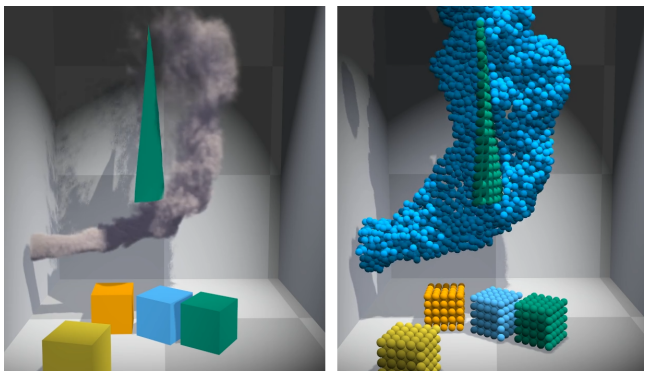

Figure 16: *Smoke interacting with cloth (left). Fluid particles are only present where smoke is located (right).*

7.2.3 Baroclinic Turbulence

Our drag model creates a velocity gradient that gives rise to some interesting motion at the fluid boundary, however it is often desirable to introduce additional turbulent motion. Fedkiw et al. [2001] introduced vorticity confinement to computer graphics in the context of smoke simulation, and Selle et al. [2005] used vortex particles to add vorticity to grid-based smoke simulations. Pfaff et al. [2012] use a mesh-based representation to model the smoke-air interface where vortex sheets are generated for each triangle and edge including a baroclinity term. [Kim et al. 2012a] use vortex particles to model baroclinic turbulence in grid-based smoke simulations.

In this section, we present a turbulence model inspired by the Lagrangian vortex sheets method [Pfaff et al. 2012]. Unlike mesh-tracking methods, we do not have an explicit surface representation, instead we use the following unnormalized vector as an approximate surface normal

$$\mathbf{n}_i = \sum_j \frac{m_j}{\rho_j} \nabla W(\mathbf{x}_i - \mathbf{x}_j). \tag{30}$$

The magnitude of this vector provides a measure of how close to the surface a particle is. Larger magnitudes indicate surface particles and smaller magnitudes indicate interior particles. Note that Eq. (30) is the standard SPH definition for the density gradient, $\nabla\rho$. We combine it with the Boussinesq pressure gradient approximation to evolve a per-particle vortex strength according to the following differential equation

$$\frac{\mathrm{D}\boldsymbol{\omega}_i}{\mathrm{d}t} = \boldsymbol{\omega}_i \cdot \nabla\mathbf{v} + \beta(\mathbf{n}_i \times \mathbf{g}), \tag{31}$$

where the first term is due to vortex stretching, and the second term represents the baroclinity. β is an exposed parameter used to control the overall strength of the effect, and the velocity gradient $\nabla\mathbf{v}$ can be approximated using SPH gradient kernels. The final driving force on a particle due to its neighbors is given by,

$$\mathbf{f}_{vort_i} = \sum_j (\boldsymbol{\omega}_j \times \mathbf{x}_{ij}) W(\mathbf{x}_{ij}). \tag{32}$$

Good results can be obtained by using a high particle drag force e.g.: $k = 50$, and allowing particle vorticity to drive the flow.

7.2.4 Rendering

Unlike grid-based solvers there is no built-in diffusion to Lagrangian schemes. To visually approximate diffusion of smoke density, we increase smoke particle size and decrease opacity over the lifetime of the particle, similar to [Brochu et al. 2012]. For our real-time renderer, we sort smoke particles according to depth from camera and render them as point sprites. Light transmission effects are approximated by rendering particles as opaque points into a shadow map and treating the light-space depth delta as input to an exponential scattering function.

8 Cloth and Ropes

We build cloth models using networks of distance constraints (Figure 17) along triangle edges to model stretching, and across edges to model bending. In addition to the basic distance constraint we expose unilateral distance constraints ("tethers") and use them as long-range attachments to reduce stretching [Kim et al. 2012b]. Cloth self-collision and inter-collision with other cloth pieces is handled automatically by the particle collision pipeline, but we note this requires sampling the cloth surface with enough particles to prevent tunneling. An aerodynamic model is applied to the cloth by approximating each triangle as a thin airfoil and distributing lift and drag forces to the triangle's particles [Keckeisen et al. 2004] .

To model ropes, we connect chains of particles with stretch and bending constraints, then extrude tessellated cylinders along their path. More advanced rope models such as [Rungjiratananon et al. 2011] may be integrated into our solver and would allow modeling of torsion. Inflatable objects can be modeled by combining a cloth mesh with a volume constraint as described in [Müller et al. 2007].

9 Implementation Details

For particle neighbor finding, we use an efficient parallel hash-grid [Green 2008]. Particle interactions are found using discrete overlap tests at the beginning of a time step, however if particles move

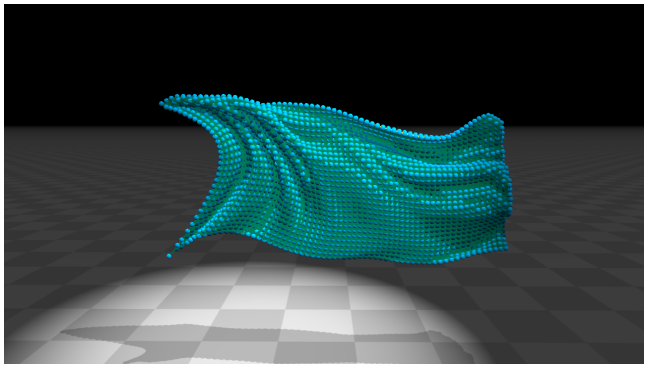

Figure 17: *A flag blowing in the wind. Self-collision is automatically handled by our particle-based collision pipeline.*

significantly during constraint solving then subsequent new collisions may be missed. To mitigate this, we allow expanding the collision radius by a fixed percentage during the overlap checks. This increases the number of potential colliders that are processed so should be set as a small fraction of the particle radius.

Constraints may be solved in a particle-centric or constraint-centric manner. In the particle-centric approach (Algorithm 2), we assign a GPU thread per particle and loop over all constraints affecting it, performing a single write operation per particle once all constraints are processed. Alternatively, we may solve constraints in a constraint-centric manner (Algorithm 3) where each thread processes a constraint and uses atomic operations to scatter position deltas to each affected particle. In practice we use both methods depending on the constraint type. For instance, fluid density constraints are solved in a particle-centric manner, and distance constraints are solved in a constraint-centric manner. Although scattered writes based on atomic memory transactions are generally less efficient, we have found this approach to be quite practical on current GPU hardware. Coherence in memory transactions is improved by re-ordering particle data according to the hash-grid cell index.

Particles are an inefficient choice of collision primitive for large shapes such as walls and floors. We represent these kinematic shapes using traditional representations such as convex hulls, signed distance fields, and triangle meshes.

10 Results

We have implemented our solver in CUDA and timings for various scenes were measured on an NVIDIA GTX680 GPU (Table 1). These times do not include rendering, however all our results were captured in real-time apart from the Rayleigh-Taylor scene which consists of 300k particles and simulates in around 150ms per frame. We render fluids using the ellipsoid splatting and screen-space surfacing described in [Macklin and Müller 2013].

Figure 1 shows fluids, rigid bodies, cloth and ropes together with two-way interactions, the drag model on our cloth slows the rigid bunnies descent via the connecting ropes. Figure 16 shows a frame from a simulation where a cloth sheet falls knocking over rigid bodies and then interacting with a smoke plume rising due to baroclinic turbulence. Figure 18 shows water balloons built from a cloth mesh filled with fluid particles. When the user pulls on the cloth mesh we dynamically remove constraints allowing them to tear the mesh and release the fluid. Because the fluid and cloth are two-way coupled, the escaping fluid forces the balloon backwards.

Algorithm 2 Particle-Centric Solve (gather)

```
for all particles i in parallel do
    initialize position delta Δx_i = 0
    for all constraints c affecting i do
        calculate constraint error λ_c, and gradient ∇_{x_i} C
        update delta Δx_i += w_i λ_c ∇_{x_i} C
    end for
end for
```

Algorithm 3 Constraint-Centric Solve (scatter)

```
for all particles i in parallel do
    initialize position delta Δx_i = 0
end for
for all constraints c in parallel do
    calculate constraint error λ_c
    for all particles i in c do
        calculate constraint gradient ∇_{x_i} C
        atomically update particle delta Δx_i += w_i λ_c ∇_{x_i} C
    end for
end for
```

Table 1: *Performance results for several examples. A frame time of 16ms is used in all cases.*

Scene	particles	diffuse	steps/frame	iters/step	ms/frame
Bunny Splash	50k	30k	2	4	10.1
Bunny Pile	44k	-	2	2	3.8
Smoke Cloth	10k	100k	2	6	5.6
Sandcastle	73k	-	2	12	10.2
Water Balloons	29k	-	3	12	12.1

11 Limitations and Future Work

In the future, we would like to remove the requirement of fixed particle sizes through the use of hierarchical acceleration structures. This would allow rigid bodies and fluids to be simulated more efficiently. Solving certain constraint types at different frequencies would also improve efficiency as in [Stam 2009].

Particles alone cannot accurately represent large flat surfaces, and our first order SDF is not accurate enough to provide useful collision normals near the surface of poorly sampled shapes. Using a quadratic basis function, as in [Corbett 2005], and reconstructing the signed distance value (instead of point sampling) could improve results for sparse samplings.

Shape-matching convergence is dependent on the total number of particles in the shape. This makes our method more suitable to smaller debris-like rigid bodies, and less suitable for large shapes. Buoyancy is also affected by the number of particles in the shape because the slower convergence makes the rigid body behave as if it were heavier. Consequently, our mass ratios do not correspond to real-world submergence depths.

12 Acknowledgments

The authors would like to thank NVIDIA and the PhysX team for supporting our work, especially Richard Tonge, Simon Schirm and Christian Sigg for many insightful discussions. We also thank the anonymous reviewers for their valuable feedback. The Bunny model is used courtesy of the Stanford Computer Graphics Laboratory.

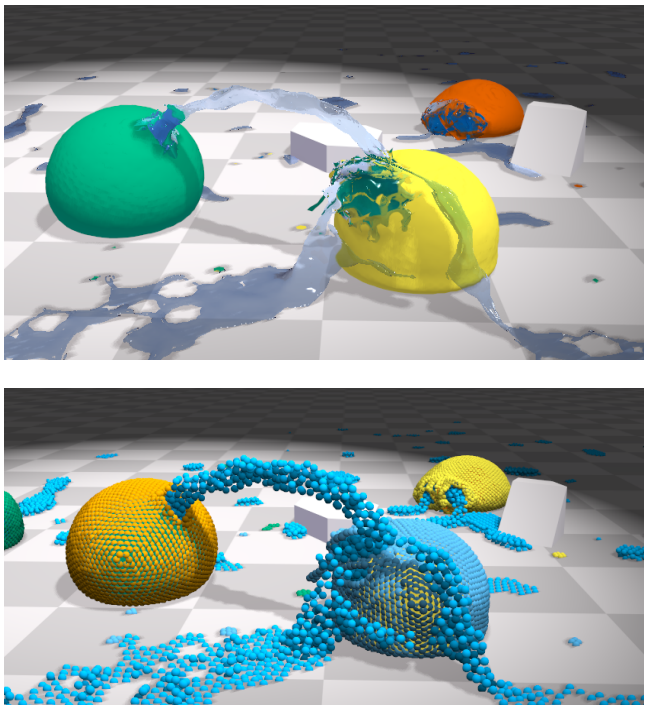

Figure 18: *A user interacts with water balloons built from a cloth mesh and filled with fluid particles under pressure. The user can tear the balloons by pulling on the cloth, releasing the fluid.*

References

AKINCI, N., IHMSEN, M., AKINCI, G., SOLENTHALER, B., AND TESCHNER, M. 2012. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph. 31*, 4 (July), 62:1–62:8.

AKINCI, N., AKINCI, G., AND TESCHNER, M. 2013. Versatile surface tension and adhesion for sph fluids. *ACM Trans. Graph. 32*, 6 (Nov.), 182:1–182:8.

AKINCI, N., CORNELIS, J., AKINCI, G., AND TESCHNER, M. 2013. Coupling elastic solids with smoothed particle hydrodynamics fluids. *Computer Animation and Virtual Worlds*.

ALDUÁN, I., AND OTADUY, M. A. 2011. Sph granular flow with friction and cohesion. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '11, 25–32.

ASCHER, U. M., CHIN, H., PETZOLD, L. R., AND REICH, S. 1995. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Structural Mechanics 23*, 2, 135–157.

BAO, Z., HONG, J.-M., TERAN, J., AND FEDKIW, R. 2007. Fracturing rigid materials. *Visualization and Computer Graphics, IEEE Transactions on 13*, 2, 370–378.

BECKER, M., IHMSEN, M., AND TESCHNER, M. 2009. Corotated sph for deformable solids. In *Proceedings of the Fifth Eurographics conference on Natural Phenomena*, Eurographics Association, 27–34.

BELL, N., YU, Y., AND MUCHA, P. J. 2005. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, SCA '05, 77–86.

BENDER, J., M"ULLER, M., OTADUY, M. A., TESCHNER, M., AND MACKLIN, M. 2014. A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum*, 1–25.

BOYD, S. P., AND VANDENBERGHE, L. 2004. *Convex optimization*. Cambridge university press.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July), 594–603.

BROCHU, T., KEELER, T., AND BRIDSON, R. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 87–95.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Transactions on Graphics (TOG)*, vol. 23, ACM, 377–384.

CARLSON, M. T. 2004. *Rigid, melting, and flowing fluid*. PhD thesis, Georgia Institute of Technology.

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, SCA '05, 219–228.

CORBETT, R. D. 2005. *Point–Based Level Sets and Progress Towards Unorganised Particle Based Fluids*. PhD thesis, The University of British Columbia.

FAURE, F. 1999. Interactive solid animation using linearized displacement constraints. In *Computer Animation and Simulation98*. Springer, 61–72.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 15–22.

GAO, Y., LI, C.-F., HU, S.-M., AND BARSKY, B. A. 2009. Simulating gaseous fluids with low and high speeds. In *Pacific Graphics 2009*, vol. 28, 1845–1852.

GASCUEL, J.-D., AND GASCUEL, M.-P. 1994. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer 10*, 4, 191–204.

GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. 2007. Efficient simulation of inextensible cloth. In *ACM Transactions on Graphics (TOG)*, vol. 26, ACM, 49.

GREEN, S. 2008. Cuda particles. *nVidia Whitepaper 2*, 3.2, 1.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. 22*, 3 (July), 871–878.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 973–981.

HARADA, T. 2007. Real-Time Rigid Body Simulation on GPUs. In *GPU Gems 3*, H. Nguyen, Ed. Addison Wesley Professional, Aug., ch. 29.

IHMSEN, M., AKINCI, N., AKINCI, G., AND TESCHNER, M. 2012. Unified spray, foam and air bubbles for particle-based fluids. *Vis. Comput. 28*, 6-8 (June), 669–677.

IHMSEN, M., WAHL, A., AND TESCHNER, M. 2012. High-resolution simulation of granular material with sph. In *Workshop on Virtual Reality Interaction and Physical Simulation*, The Eurographics Association, 53–60.

JAKOBSEN, T. 2001. Advanced character physics. In *Game Developers Conference*, 383–401.

KECKEISEN, M., KIMMERLE, S., THOMASZEWSKI, B., AND WACKER, M. 2004. Modelling effects of wind fields in cloth animations.

KIM, D., LEE, S. W., YOUNG SONG, O., AND KO, H.-S. 2012. Baroclinic turbulence with varying density and temperature. *IEEE Transactions on Visualization and Computer Graphics 18*, 1488–1495.

KIM, T.-Y., CHENTANEZ, N., AND MÜLLER-FISCHER, M. 2012. Long range attachments - a method to simulate inextensible clothing in computer games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 305–310.

MACKLIN, M., AND MÜLLER, M. 2013. Position based fluids. *ACM Trans. Graph. 32*, 4 (July), 104:1–104:12.

MARTIN, S., KAUFMANN, P., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2010. Unified simulation of elastic rods, shells, and solids. In *ACM SIGGRAPH 2010 Papers*, ACM, New York, NY, USA, SIGGRAPH '10, 39:1–39:10.

MÜLLER, M., AND CHENTANEZ, N. 2011. Adding physics to animated characters with oriented particles. In *Workshop in Virtual Reality Interactions and Physical Simulation*, The Eurographics Association, 83–91.

MÜLLER, M., AND CHENTANEZ, N. 2011. Solid simulation with oriented particles. In *ACM Transactions on Graphics (TOG)*, vol. 30, ACM, 92.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '03, 154–159.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 141–151.

MÜLLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004. Interaction of fluids with deformable solids. In *JOURNAL OF COMPUTER ANIMATION AND VIRTUAL WORLDS (CAVW*, 159–171.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, SIGGRAPH '05, 471–478.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr.), 109–118.

PARK, S. I., AND KIM, M. J. 2005. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 261–270.

PFAFF, T., THUEREY, N., AND GROSS, M. 2012. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph. 31*, 4 (July), 112:1–112:8.

RUNGJIRATANANON, W., KANAMORI, Y., METAAPHANON, N., BANDO, Y., CHEN, B.-Y., AND NISHITA, T. 2011. Twisting, tearing and flicking effects in string animations. In *Motion in Games*. Springer, 192–203.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost sph for animating water. *ACM Transactions on Graphics (TOG) 31*, 4, 61.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. 24*, 3 (July), 910–914.

SOLENTHALER, B., AND PAJAROLA, R. 2008. Density contrast sph interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '08, 211–218.

SOLENTHALER, B., SCHLÄFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid–solid interactions: Research articles. *Comput. Animat. Virtual Worlds 18*, 1 (Feb.), 69–82.

STAM, J., AND FIUME, E. 1995. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 129–136.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 121–128.

STAM, J. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics' 09. 11th IEEE International Conference on*, IEEE, 1–11.

TONGE, R., WYATT, B., AND NICHOLSON, N. 2010. Physx gpu rigid bodies in batman: Arkham asylum. *Game Programming Gems 8*, 590–601.

TONGE, R., BENEVOLENSKI, F., AND VOROSHILOV, A. 2012. Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph. 31*, 4 (July), 105:1–105:8.

WEINSTEIN, R., TERAN, J., AND FEDKIW, R. 2006. Dynamic simulation of articulated rigid bodies with contact and collision. *Visualization and Computer Graphics, IEEE Transactions on 12*, 3, 365–374.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, SIGGRAPH '05, 965–972.