Constraint-Based Simulation of Interactions Between Fluids and Unconstrained Rigid Bodies

Sho Kurose*
The University of Tokyo, Japan.

Shigeo Takahashi†
The University of Tokyo, Japan.

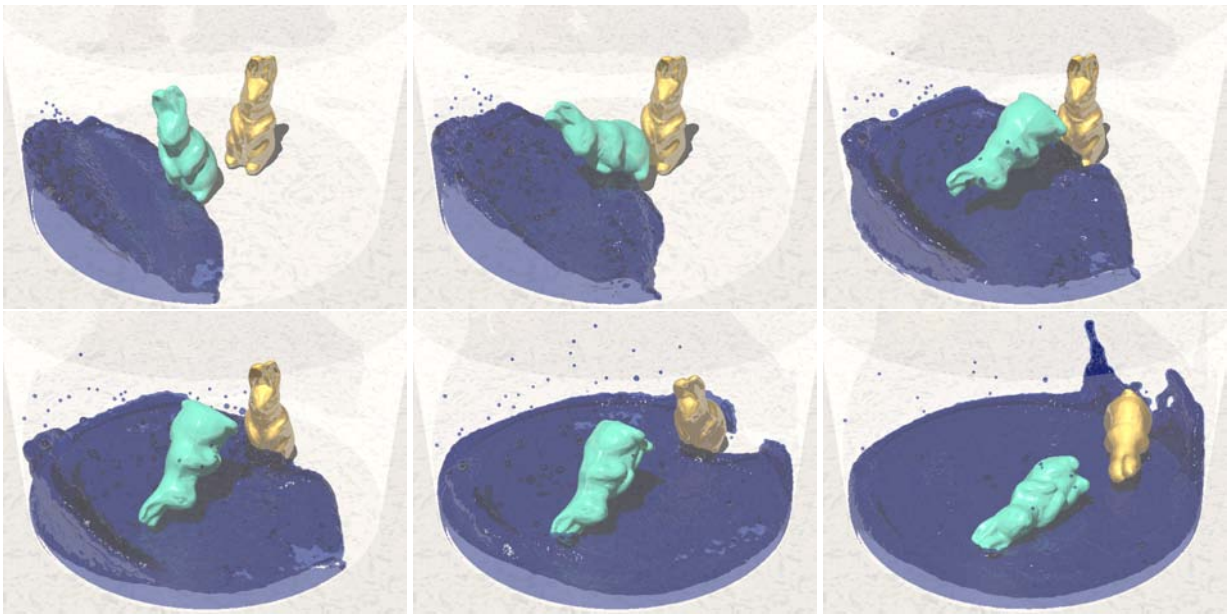


Figure 1: Simulating interactions between fluid and multiple rigid bodies with our approach.

Abstract

We present a method for simulating stable interactions between fluids and unconstrained rigid bodies. Conventional particle-based methods used a penalty-based approach to resolve collisions between fluids and rigid bodies. However, these methods are very sensitive to the setting of physical parameters such as spring coefficients, and thus the search for appropriate parameters usually results in a tedious time-consuming task. In this paper, we extend a constraint-based approach, which was originally developed for calculating interactions between rigid bodies only, so that we can simulate collisions between fluids and unconstrained rigid bodies without worrying about the parameter tweaking. Our primary contribution lies in the formulation of such interactions as a linear complementary problem in such a way that it can be resolved by straightforwardly employing Lemke's algorithm. Several animation results together with the details of GPU-based implementation are presented to demonstrate the applicability of the proposed approach.

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 and Virtual reality

Keywords: physically-based simulation, constraint-based, fluids, rigid body, linear complementarity problem

*e-mail: kurose@visual.k.u-tokyo.ac.jp
†e-mail: takahashis@acm.org

1 Introduction

Simulating interactions between fluids and rigid bodies has become popular in computer games and movie films. Realistic behaviors of fluids and rigid bodies give us the feeling of being at the scenes. When we try to generate such realistic scenes, however, we have to adjust a large number of physical parameters, which inevitably results in a tedious time-consuming process. This leads us to the need for a more sophisticated model for simulating such interactions, which prevents us from having troubles in exploring acceptable physical parameters.

Particle-based methods have been effectively used for simulating the interactions between fluids and rigid bodies because it can easily calculate how the rigid bodies will respond to local motions of the water surface flow. However, in practice, such conventional particle-based methods require careful tweaking of physical parameters for visually-acceptable simulation results, due to their poor approximation of collisions between fluids and rigid bodies.

This paper presents a method for simulating accurate interactions between fluids and rigid bodies, by taking advantage of a constraint-based formulation [Baraff 1989; Baraff 1994]. Our method allows

us to avoid a time-consuming trial and error process for setting appropriate parameters because the associated simulation can be carried out in a stable manner while allowing for relatively large time steps. Nonetheless the constraint-based formulation has originally been developed for calculating interactions between rigid bodies only, we extend this to handle collisions between fluids and rigid bodies in this study.

In the constraint-based method, a *Linear Complementarity Problem* (LCP) must be solved to compute contact forces. Lemke's algorithm [Lemke 1965] is an efficient solver for the LCP, however, the algorithm leads to unstable computation especially when rigid bodies have many contact points at a time during collision as shown in Figure 2. We resolve this problem by representing each fluid particle as a point mass, because, in this setting, a fluid particle collides with a rigid body only at a single contact point. Furthermore, together with the assumption that all the rigid bodies are unconstrained, we can employ Lemke's algorithm straightforwardly in order to simulate interactions between fluids and rigid bodies in a stable manner. The enhancement of this formulation can be implemented on GPUs, where more than 1,200 contact points at a time can be successfully resolved (See Figure 7 for example.).

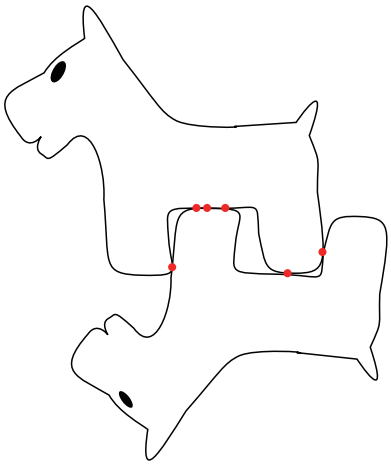


Figure 2: Rigid bodies colliding with each other on multiple contact points. Contact points are represented in red.

The remainder of this paper is organized as follows. After referring to related work for interactions between fluids and rigid bodies in Section 2, we describe the fluid and rigid body models employed in our approach in Section 3. Section 4 presents an algorithm for simulating the interactions between fluids and unconstrained rigid bodies using the constraint-based method. In Section 5, the GPU implementation of the present algorithm is introduced for its accelerated computation. Section 6 demonstrates experimental results to clarify the applicability of our approach. Finally, we conclude this paper and refer to future work in Section 7.

2 Related Work

This section provides a survey on existing methods for coupling between fluids and rigid bodies in physically-based simulation. If rigid bodies are constrained in their motion in some way, they are defined to be constrained rigid bodies. A chain of rigid bodies connected by rotational joints is such an example. To the best of our knowledge, none has been done for simulating the interactions of such constrained rigid bodies with fluids, and thus all the conventional methods were developed for the interactions between fluids and unconstrained rigid bodies. Basically, these methods can be classified into three categories: *one-way solid-to-fluid* coupling, *one-way fluid-to-solid* coupling, and *two-way* coupling.

In the *one-way solid-to-fluid* coupling, we only consider force transition from solid objects to fluids. Simulations of a water splash generated by a ball thrown into a water tank [Foster and Metaxas 1997; Foster and Fedkiw 2001; Enright et al. 2002] and a paddle wheel rotating through fluids [Batty et al. 2007] are examples of this category.

On the other hand, the *one-way fluid-to-solid* coupling only takes into account inverse forces produced by fluids to rigid bodies. Foster et al. [1996] demonstrated this type of coupling by simulating tin cans floating on top of swelling water. Yuksel et al. [2007] animated many boats drifting among ocean waves using this type of coupling.

In the *two-way* coupling of fluids and rigid bodies, they respond to each other when the collision occurs between them. This paper focuses on an approach based on this *two-way* coupling.

The development of the two-way coupling methods began by employing the Eulerian grid-based simulation scheme. Takahashi et al. [2002] presented a simple method for coupling between fluids and buoyant rigid bodies on regular grids using a combined *Volume of Fluid* method and *Cubic Interpolated Propagation* system. However, their technique poorly approximates interactions between fluids and rigid bodies because it neglects the dynamic forces and torques due to the fluid momentum. Génevaux et al. [2003] used marker particles for a free surface fluid simulation and demonstrated a two-way coupling of fluids and deformable solids represented as spring-mass conglomerations. However, this method cannot be applied easily to non-deformable rigid bodies having complex shapes. Carlson et al. [2004] performed the two-way coupling of fluids and rigid bodies using the *Distributed Lagrange Multipliers*, which approximates rigid bodies as fluids on a grid. This method projects the velocity field inside rigid bodies back to rigid motion, while this incurs discontinuities in the velocity field and thus results in unexpected leaks of fluids through rigid bodies. Moreover, the method cannot simulate the interactions between fluids and rigid bodies in a stable manner especially when the densities of the rigid bodies are less than those of the fluids. This is due to the fact that the force caused by the collisions between rigid bodies depends on the difference in density between the rigid bodies and fluids.

In general, these Eulerian grid-based methods suffer from handling local motions of fluid flows such as water splashes because it cannot faithfully represent the local shapes of water surfaces. Lagrange particles have been introduced to alleviate this problem and successfully simulated the complex spatio-temporal behaviors of water surfaces. Using the *Smoothed Particle Hydrodynamics* (SPH) method [Monaghan 1992; Müller et al. 2003] for the fluid dynamics, Müller et al. [2004] demonstrated interactions between particle-based fluids and mesh-based deformable solids, by representing the solid as a tetrahedral mesh and distributing particles around them. However, in this work, the associated simulation results are not precise enough to produce visually-plausible animations. A similar method [Amada et al. 2004] has also been proposed that represents each rigid body as a set of particles in order to detect and resolve collisions between fluids and rigid bodies efficiently. In addition, Tanaka et al. [2007] used the Moving Particle Semi-implicit (MPS) method [Koshizuka and Oka 1996; Premože et al. 2003] instead of the SPH method. In their approach, the MPS method is more efficient than the SPH method because they represented rigid bodies with lattice particle positions.

In practice, all of these methods employ a penalty-based approach to resolve collisions between fluids and rigid bodies. Although the penalty-based approach is easy to implement and its computation complexity is linear with respect to the number of collisions, the stability of its associated simulation highly depends on the stiffness and damping parameter values that govern the contact forces between fluids and rigid bodies. However, adjusting these parameters to avoid unnatural self-intersections has still been a daunting task.

Solenthaler et al. [2007] used the SPH method for the simulation of liquids and deformable objects as well as rigid objects. They used the SPH-based method to compute the pressure and viscosity forces that have influence on the particles. In their framework, these forces are handled as contact forces that arise in the collisions among liquids, deformable objects, and rigid objects. Avoiding interpenetration between these objects, however, inevitably needs careful adjustment of the gas constant as well as the stiffness parameter for the penalty-based method, and thus results in crude approximation of their collisions.

Quite recently, Becker et al. [2009] have presented a constraint-based method for resolving collisions between fluids and unconstrained rigid bodies. Instead of solving the LCP, they formulated the constraint equation for the relative velocity between a fluid particle and a single rigid body at their contact point using the coefficient of restitution, and then computed the total forces and torques accumulated on the rigid body. Their approach is advantageous in that the equation is always defined as a linear 6×6 system of equations, no matter how large the number of associated collisions is. However, the approach can handle the interactions only between fluids and a single rigid body, and cannot handle the case where multiple rigid bodies collide with each other.

We employ the constraint-based method in order to avoid troublesome trial and error processes, which the penalty-based method imposes on us. To simulate interactions between fluids and unconstrained multiple rigid bodies, we solve the LCP to compute a contact force at a contact point.

3 Fluid and Rigid Body Models

In this section, we explain the fluid and rigid body models employed in our approach.

3.1 Fluid Model

A smoothed particle hydrodynamics (SPH) method has become the most popular particle-based method for simulating fluid dynamics. The SPH method was originally developed by Lucy [1977] and Gingold et al. [1977] for the simulation of nonaxisymmetric phenomena in astrophysics. According to the SPH formulation, a physical quantity A at the position $\mathbf{x}$ is interpolated by blending those of the particles in the local neighborhood only, which is formulated as:

$$A(\mathbf{x}) = \sum_i m_i \frac{A_i}{\rho_i} W(\mathbf{x} - \mathbf{x}_i, h), \tag{1}$$

where m_i, ρ_i and $\mathbf{x}_i$ are the mass, density, and position of the i-th particle, respectively. $W(\mathbf{x}, h)$ is a weight function called a smoothing kernel with the core radius h.

A conventional particle-based method in [Müller et al. 2003] computes various forces that influence on the velocities of particles. However, such an explicit scheme is likely to be unstable because it is sensitive to the size of the time step. In order to prevent such instability, Clavet et al. [2005] introduced an implicit scheme called *Prediction-Relaxation Scheme* to the SPH approach. This implicit scheme directly computes the displacements and velocities of particles, instead of forces applied to them. Note that the local displacement and velocity are separately used to update the particle positions step by step, in order to maximally preserve the stability of the associates simulation. In this approach, we use Clavet et al.'s method especially for computing the fluid dynamics.

3.2 Rigid Body Model

In general, polygonal mesh representation is frequently used to represent the shapes of rigid bodies, while it requires intricate algorithms for detecting collisions with fluids, which may result in a high computational cost in our framework. In this approach, a rigid body is modeled as a set of particles in order to systematically detect and resolve collisions with particle-based models of fluids. We use Crane et al.'s approach [2007] for approximating rigid bodies with particles, as shown in Figure 3.

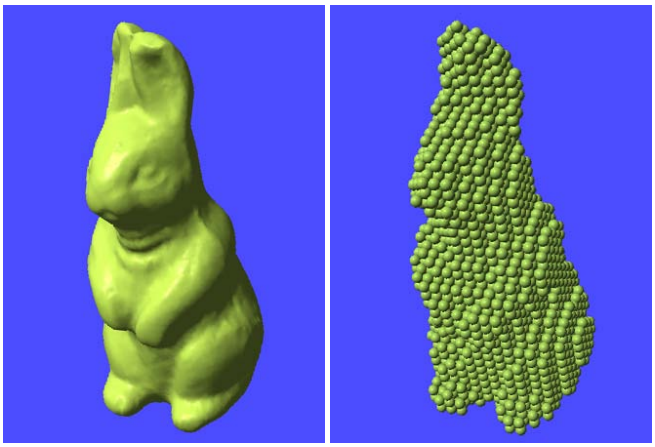


Figure 3: Polygonal representation of a rabbit model and its particle-based representation. The rabbit model on the right consists of 3,352 particles.

4 Constraint-Based Interactions

This section presents an algorithm for simulating interactions between fluids and rigid bodies using the constraint-based method.

4.1 Detecting Collision

For handling collisions between fluids and rigid bodies, our first task is to detect whether rigid bodies are colliding with fluids. Since fluids and rigid bodies are composed of particles as described in Section 3, we can write the conditions for finding such collisions as:

$$\|\mathbf{X}_f - \mathbf{X}_r\| \leq r_f + r_r, \text{ and} \tag{2}$$

$$\mathbf{n} \cdot (\mathbf{V}_f - \mathbf{V}_r) \leq 0, \tag{3}$$

where $\mathbf{X}_f$ and $\mathbf{X}_r$ denote the positions of the fluid and rigid particles, r_f and r_r represent their radii. $\mathbf{n}$ is the unit surface normal of a rigid body at the contact point, and $\mathbf{V}_f$ and $\mathbf{V}_r$ are the velocities

of fluid and rigid particles, respectively. Figure 4 shows such a collision between the fluid and rigid particles. Here, $\mathbf{V}_r$ is expressed as $\mathbf{V}_r = \mathbf{V}_{par} + \omega \times \mathbf{r}$, where $\mathbf{V}_{par}$ and ω are the linear and angular velocities of the corresponding rigid body, respectively. Eq. (2) implies that the fluid and rigid particles are overlapping at the contact point $\mathbf{X}_p = \mathbf{X}_r + r_r\mathbf{n}$. Eq. (3) shows that the relative velocity of the fluid particle with respect to the rigid particle along the normal direction is less than zero. If the relative velocity is positive, the particles will be no longer interpenetrated because they will be separated in a future time step.

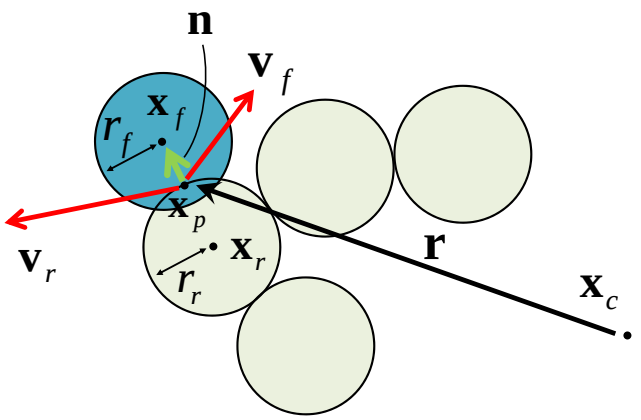


Figure 4: A collision between fluid and rigid particles

Calculating the possible collision between every pair of particles is computationally expensive even for a small number of particles. We avoid this expensive computation by partitioning the 3D space into small cube-like regions called *cells* on a 3D grid, where the size of the regions is set to be the largest particle diameter among the particles representing fluids and rigid bodies. For each particle, we will then collect cells that are adjacent to the cell that contains the target particle, and check the collisions of the target particle with those contained in these cells only. This allows us to efficiently reduce the computational cost by minimizing the number of cells we have to visit for the collision detection. In our implementation, the diameters of rigid particles are equal to those of fluid particles, and thus we only visit 26 neighboring cells in addition to the cell that contains the target particle since the cells are aligned with the 3D grid.

4.2 Resolving Collisions

In our approach, we treat collisions between fluids and rigid bodies as those between rigid bodies only by representing a fluid particle as a point mass. For resolving these collisions, we have to compute contact forces between fluids and rigid bodies that preclude the interpenetration of object surfaces. The most popular approach for modeling such contact forces is a penalty-based method, which defines the contact force $\mathbf{F}$ as $\mathbf{F} = k_s d\mathbf{n} + k_d(\mathbf{v}\cdot\mathbf{n})\mathbf{n}$, where d is the amount of the interpenetration between a pair of colliding rigid bodies A and B, $\mathbf{n}$ is a unit surface normal of B, $\mathbf{v}$ is the relative velocity of A with respect to B, and k_s and k_d are stiffness and damping parameters, respectively. The stiffness parameter controls the strength of the force to avoid interpenetration and must be large enough to avoid any undesirable visual artifacts. However, it is difficult to adjust this parameter value systematically and often requires a tedious trial and error process for searching an proper value. Similarly, tweaking of the damping parameter will also require special attention. Another problem with the penalty-based method is that it expects very small time steps for stably updating the temporal snapshot of the simulation. This means that the penalty-based method is not appropriate for resolving collisions between fluids and rigid bodies especially when the associate configuration of objects is relatively complicated because it is prone to multiple collisions at a time.

An impulse-based method is another way to handle the collisions. Mirtich et al. [1995] employed the method to simulate interactions between rigid bodies, and Bridson et al. [2002] extended the method to handle deformable models. The impulse-based methods do not compute a force that influences on the accelerations of colliding rigid bodies, but an impulse that connects to their velocities. This enables us to update the velocities of the rigid bodies without explicitly integrating their acceleration with respect to the time. In practice, the impulse-based method is more stable than the penalty-based method because it just requires us to control a coefficient of restitution ε for elasticity, which is much easier to be adjusted than the stiffness and damping parameters in the penalty-based method. A major limitation of the impulse-based methods is that it assumes the occurrence of each collision to be isolated in space and time, i.e., every collision happens at one contact point at a time. This implies that the impulse-based method cannot fully simulate the complicated configuration of fluids and rigid bodies since it will readily contains simultaneous multiple contact points between them.

This simultaneous multiple contacts problem can be solved using constraint-based methods [Baraff 1989; Baraff 1994]. The constraint-based method provides a high degree of accuracy. For each contact, the constraint-based method defines a non-penetration constraint. These defined constraints can be formulated as a LCP. In practice, the constraint-based method solves this LCP to compute contact forces that prevent such interpenetration. We employ the constraint-based method to resolve collisions between fluids and rigid bodies, while representing a fluid particle as a point mass.

4.2.1 Identifying the type of collision

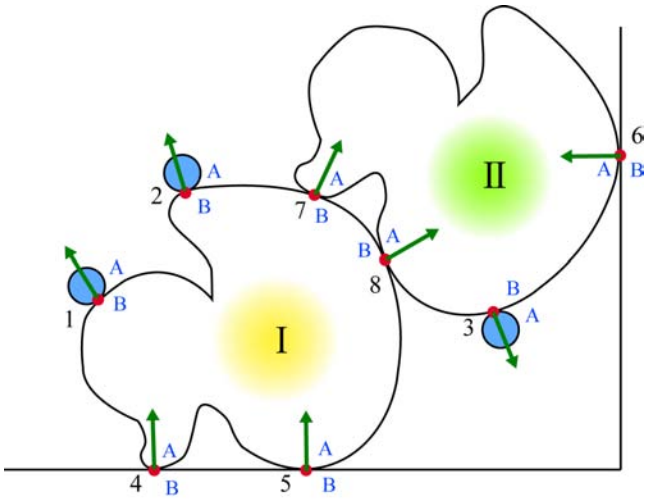


Figure 5: Simultaneous multiple contacts between fluid and rigid bodies. Red points indicate the contact points. Each green arrow represents the normal direction at the contact point. The points 1, 2, and 3 are contacts between fluid particles and rigid bodies, 4, 5, and 6 are contacts between rigid bodies and a wall, and 7 and 8 are contacts between the two rigid bodies.

In our approach, we first identify each contact point as one of the

three types as shown in Figure 5: a contact between fluid and an unconstrained rigid body, a contact between an unconstrained rigid body and a non-movable rigid body (such as a wall), and a contact between two unconstrained rigid bodies. Solving the aforementioned LCP amounts to computing contact forces at these contact points appropriately by referring to the associated types of collisions. In our framework, we must collect the following information for each contact point:

- the position of the contact point **p**,
- the IDs of the two colliding objects A and B, and
- the normal at the contact point **n**.

For the contact between fluid and an unconstrained rigid body (i.e. the first type of collision), we assume that A and B are the IDs of the fluid and rigid body, respectively. For the contact between unconstrained and non-movable rigid bodies (i.e. the second type of collision), A is the ID of the non-movable rigid body while B is the unconstrained one. When A and B are both unconstrained rigid bodies, we assign a smaller ID to B. Figure 5 shows how we set the object IDs to A and B.

Furthermore, the normal at the contact point is assumed to be the unit surface normal of the object B. Note that we uniquely assign object IDs to A and B this way because we want to avoid bothering with the choice of surface normal directions (i.e. the signs of the normal vectors **n**) in our formulation.

4.2.2 LCP formulation

Suppose that we have k simultaneous collisions at time t. In this situation, we can find an impulse $\mathbf{J}_i = f_i\mathbf{n}_i$ at a contact point i ($1 \leq i \leq k$), where $\mathbf{n}_i$ is the normal vector at the contact point and f_i is the i-th entry of a vector of unknown impulse magnitudes $\mathbf{f}$. The goal of resolving these collisions is to find an appropriate vector $\mathbf{f}$ that keeps the fluids and rigid bodies from interpenetrating.

According to [Baraff 1989], we can compute an appropriate vector $\mathbf{f}$ if all the contact forces satisfy the following non-penetration constraints:

(1) The impulse does not allow colliding objects to interpenetrate.

(2) The impulse only applies forces to the colliding objects so that they become apart from each other.

(3) The impulse becomes zero at the corresponding contact point once the two colliding objects begin to come apart.

Now, at time t, let v_i^t be the relative velocity of the object A with respect to the object B in the $\mathbf{n}_i$ direction at the contact point i. Similarly, let $v_i^{t+\Delta t}$ be the corresponding relative velocity at time $t+\Delta t$. Here, Δt denotes a small time step interval. With this notation, we can write the non-penetration constraints as

$$v_i^{t+\Delta t} + \varepsilon v_i^t \geq 0, \tag{4}$$
$$f_i \geq 0, \text{ and} \tag{5}$$
$$f_i(v_i^{t+\Delta t} + \varepsilon v_i^t) = 0, \tag{6}$$

where ε is a coefficient of restitution.

These constraints (4), (5), and (6) can be reduced to the LCP formulation. Formally speaking, the LCP is defined as the following problem of finding a vector $\mathbf{x} \in \mathbf{R}^k$ such that

$$\mathbf{Ax} + \mathbf{b} \geq 0, \tag{7}$$
$$\mathbf{x} \geq 0, \text{ and} \tag{8}$$
$$\mathbf{x}^T(\mathbf{Ax} + \mathbf{b}) = 0, \tag{9}$$

where $\mathbf{A} \in \mathbf{R}^{k\times k}$ and $\mathbf{b} \in \mathbf{R}^k$.

Before solving this LCP problem with a numerical solver, we must compute the matrix $\mathbf{A}$ and vector $\mathbf{b}$. These values can be obtained from the correspondence between the equations (4) and (7). Since each impulse at some contact point will influence on the forces at all the other contact points through the colliding objects, $v_i^{t+\Delta t}$ can be defined as a linear combinations of f_i's. Due to this, we can rewrite the equation (4) as follows:

$$\sum_{j=1}^{n} f_j a_{ij} + v_i^t(1+\varepsilon) \geq 0. \tag{10}$$

Here,

$$v_i^t(1+\varepsilon) = b_i, \tag{11}$$

a_{ij} denotes the (i,j)-entry of the matrix $\mathbf{A}$, and b_i is the i-th element of the vector $\mathbf{b}$, respectively. Note that the matrix entry a_{ij} shows how the impulse at the contact point j affects the impulse at the contact point i.

Since the impulse depends on the force and torque acting on the object, a_{ij} can be expressed as follows:

$$a_{ij} = \mathbf{n}_i \cdot \left(\left(\frac{\mathbf{n}_j}{M_{A_i}} + I_A^{-1}(\mathbf{r}_A \times \mathbf{n}_j) \right) - \left(\frac{-\mathbf{n}_j}{M_{B_i}} + I_B^{-1}(\mathbf{r}_B \times (-\mathbf{n}_j)) \right) \right). \tag{12}$$

Here, M_{A_i} and M_{B_i} are the masses of objects A and B for the contact point i, $\mathbf{n}_i$ and $\mathbf{n}_j$ are the normals of the contact points i and j, I_A^{-1} and I_B^{-1} are the inverse inertia tensors of the colliding objects A and B, and $\mathbf{r}_A$ and $\mathbf{r}_B$ are the vectors emanating from the centers of objects A and B to the contact point, respectively. As described in Section 4.2.1, we can use this equation straightforwardly, without worrying about the orientations of the normals at the contact points.

Although we can get the matrix $\mathbf{A}$ and vector $\mathbf{b}$ from Eqs. (11) and (12), several issues should be noted when we get the matrix $\mathbf{A}$ from Eq. (12). The inertia tensor of the fluid particle is zero because we represent a fluid particle as a point mass. Non-movable objects, such as walls and floors, are usually assumed to have infinite mass. This implies that we have to set the inverse of the object mass to be zero. Furthermore, if the contact points i and j are not involved in the same object (e.g. the contact points 1 and 3 in Figure 5), the corresponding matrix entry a_{ij} is zero because the contact point i has no influence on the contact point j, and vice versa. According to Baraff [1989], in a friction-free system, the formulated matrix $\mathbf{A}$ is guaranteed to be positive semi-definite (PSD). The LCP including the PSD matrix can be solved by Lemke's algorithm, which is an efficient LCP solver. We assume such a frictionless system in our framework so that we can take full advantage of Lemke's algorithm in order to simulate visually plausible interactions between fluids and rigid bodies.

In practice, we can consider special cases that violate the above assumption of a friction-free system. For example, highly viscous fluids incur friction forces when they stick to rigid bodies. Clavet et al. successfully simulated such phenomena in their framework [Clavet et al. 2005] by applying virtual attraction forces between the fluids and rigid bodies to make them stay close to each other, once

they resolved their collisions. Their technique allows us to simulate frictional systems also in our framework while its use is limited to dynamic friction and cannot be extended to the simulation of static friction forces.

The impulse magnitudes calculated through Lemke's algorithm will then be applied to appropriately update the temporal snapshots of objects' velocities and momentum in the simulation.

5 GPU Implementation

We enhanced our method through a GPU implementation, by taking advantage of Harada et al.'s GPGPU technique [Harada et al. 2007], which uses a texture buffer in the video memory to search for neighbors of a target particle. The texture consists of a 2D grid of pixels, each of which has red, green, blue, and alpha (RGBA) channels. By using a shading language such as GLSL, each channel can store an arbitrary value. In their method, each pixel of the texture image has been matched with a cell described in Section 4.1, and the corresponding four channels were used to store the indices of neighboring particles to be searched. Harada et al. used only one texture image for describing neighborhood relationships between particles since they could simulate realistic behaviors of particles by visiting four neighboring particles only, while we introduced more texture images to visit more particles so that we can fully simulate viscoelasiticity of the fluids inherent to Clavet et al.'s framework [Clavet et al. 2005]. In our implementation, we employed three texture images in order to keep up to 12 neighbors for each particle.

We also apply this technique to accelerate the collision detection between fluids and rigid bodies. Although we have to read data back from the GPU so that we can handle the LCP computation on the CPU, we still achieved a speedup by a factor of more than 12, compared to the equivalent CPU implementation.

More specifically, we store the indices of fluid and rigid body particles in each cell in the GPU so as to search for neighboring and colliding particles. The GPU is also responsible for computing the fluid flows based on the SPH and updating the positions of fluid particles. On the other hand, the CPU computes the matrix $\mathbf{A}$ and vector $\mathbf{b}$ in the LCP (Eq. (7)) at every time step, and resolves the collisions among fluid and rigid body particles using Lemke's method. The data transmitted from the GPU to the CPU contains the IDs of the colliding objects the positions of the contact points, the surface normals at the contact points, and the relative velocities in the normal directions. Conversely, the GPU receives from the CPU the impulses that influence on the velocities of fluid particles, together with the updated positions and velocities of the rigid bodies.

6 Results and Discussions

All experiments in this paper were conducted on a PC with an Intel Core 2 Duo 2.13GHz processor, 2.0GB RAM, and an NVIDIA GeForce 8800 GT graphics card with 1.0GB video RAM. The software was implemented in C++, OpenGL and GLSL. In our implementation, we performed the simulation using point-based rendering at the first stage, and then improved the rendering quality as a post-process with the POV-Ray software. Note that the statistics on frame rates in this paper exclude this post-process stage of rendering.

Figure 1 shows interactions with fluids and two rigid bodies having different densities. Note that the densities of the green and golden rabbits are 0.5 and 5 times larger than that of the fluid. Each rabbit is sampled with 1,710 particles, and the number of fluid particles is 82,944. This animation is 4.3 seconds long and the average frame rate on the GPU is 6.4 fps and the time interval (Δt) is 1.0 msec in the animation. The average frame rate on the CPU only is 0.36 fps. The maximum number of simultaneous contact points is 464. This result demonstrated that our scheme can successfully visualize plausible interactions between fluid and rigid bodies while fully taking into account the differences in their densities. On the other hand, conventional penalty-based approaches require time-consuming trial and error processes to seek appropriate stiffness and damping parameters for each rigid body.

In Figure 6, a toy duck is pushed by water spraying from a fountain. In this scene, we used 41,616 particles for the entire fluid and 917 particles for the toy duck. This simulation was 5.0 seconds long. Its average frame rate on the GPU was 22.7 fps and that on the CPU was 1.90 fps, while the corresponding time interval (Δt) was 1.5 msec. The maximum number of simultaneous contact points was 41 in this case. As shown in this figure, we can simulate realistic spatio-temporal behavior of the rigid body in response to local motions of the water surface flow.

We also simulated a large number of collisions between fluid and a complicated rigid body as shown in Figure 7. In this 7.0 seconds simulation, 262,144 fluid particles and 6,432 rigid particles were used. The average frame rate on the GPU was 1.5 fps for the simulation, and that on the CPU only was 0.041 fps. The time interval (Δt) employed here was 1.0 msec. The maximum number of simultaneous contact points for this scene was 1,249. This result indicates that our scheme can handle such a large number of simultaneous collisions. In Table 1, we summarize these statistics for three example scenes.

Our scheme can also simulate accurate interactions between fluids, unconstrained rigid bodies, and non-movable objects such as walls and floors. Figure 8 shows the comparison between the simulation results with the conventional penalty-based method and our method. We can notice from the figure that the teapot interpenetrates the wall and the fluid also interpenetrates the teapot using the penalty-based method while this unexpected artifact is successfully avoided using our method. Although the impulse-based methods [Mirtich and Canny 1995; Bridson et al. 2002] may achieve similar results, they are highly likely to be unstable when the given configuration of fluids and rigid bodies is too complicated.

7 Conclusion and Future Work

We have presented an approach to simulating interactions between fluids and unconstrained rigid bodies. We extend the conventional constraint-based approach to model collisions between fluids and rigid bodies by representing a fluid particle as a point mass. This extension allows us to avoid a time-consuming trial and error process for setting appropriate parameters and to take full advantage of Lemke's algorithm for computing contact forces between the fluids and rigid bodies. Several animation results have been presented to demonstrate that our scheme can simulate various cases, where the fluid and rigid bodies intricately collide with each other.

In this paper, frictional interactions between fluids and rigid bodies were not taken into account. When we introduce the effects of friction contacts between the colliding objects, the matrix $\mathbf{A}$ in the LCP (Eq. (7)) may violate the conditions of the PSD. For the solution of such type of LCP, we have to improve the associated Lemke's algorithm. Furthermore, we can still enhance our computation for finding the nearest particles by implementing avilable k-D

Scene	Fluid Particles	Avg. FPS (GPU / CPU)	Time Interval	Max Simultaneous Contact Points
Fountain	41,616	22.7 / 1.90	1.5	41
Multiple Bodies	82,944	6.4 / 0.36	1.0	464
Large Contacts	262,144	1.5 / 0.041	1.0	1,249

Table 1: Statistics for example scenes.

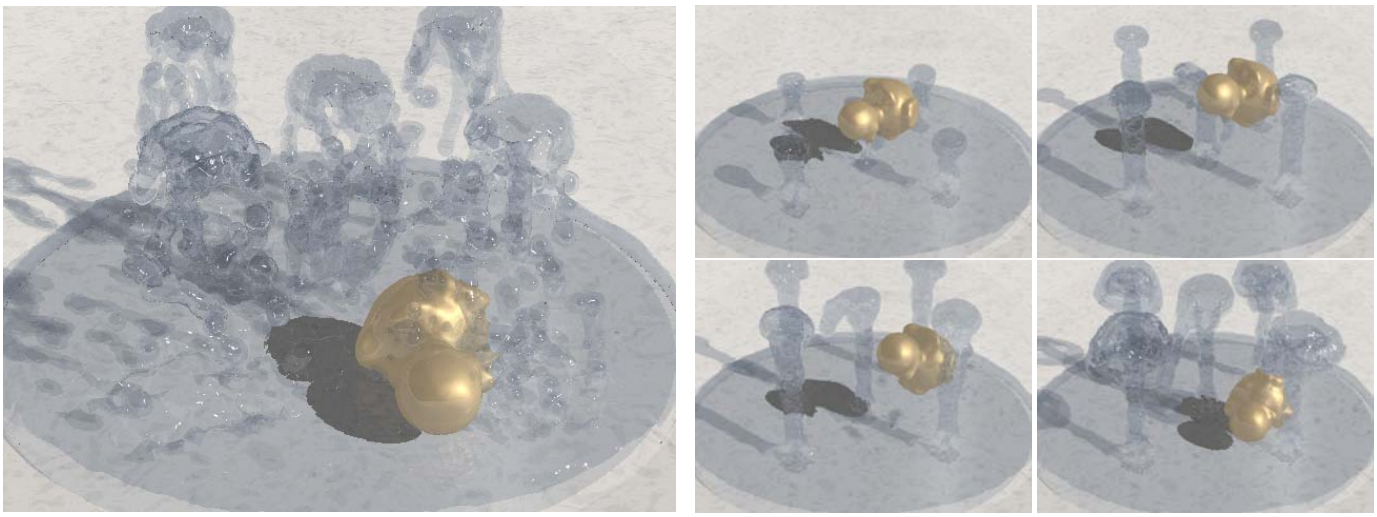
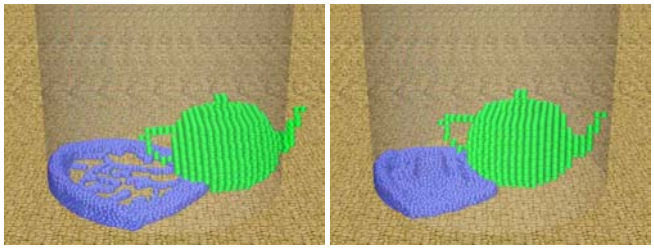


Figure 6: A fountain simulation.

tree algorithms on GPU [Horn et al. 2007]. We would also like to extend our method to handle interactions of multiple types of fluids together with rigid bodies.

Acknowledgements

We would like to thank Satoshi Mabuchi for his assistance in programe implementation, and anonymous reviewers for their valuable comments. This work was partially supported by Grants-in-Aid for Scientific Research (B) (20300033).



(a) (b)

Figure 8: Comparison between the simulation results with (a) the penalty-based method and (b) the present method. The penalty-based method cannot avoid the interpenetrations among the fluid, the teapot and the wall.

References

AMADA, T., IMURA, M., YASUMURO, Y., MANABE, Y., AND CHIHARA, K. 2004. Particle-based fluid simulation on GPU. In *Proc. ACM Workshop on General-Purpose Computing on Graphics Processors*.

BARAFF, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proc. ACM SIGGRAPH '89*, 223–232.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. ACM SIGGRAPH '94*, 23–34.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics 26*, 100.

BECKER, M., TESSENDORF, H., AND TESCHNER, M. 2009. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 99*, 2.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics*, 594–603.

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

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005*, 219–228.

CRANE, K., LLAMAS, I., AND TARIQ, S. 2007. Real-time simulation and rendering of 3D fluids. In *GPU Gems 3*, H. Nguyen, Ed. Addison Wesley, ch. 30, 633–675.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics*, 736–744.

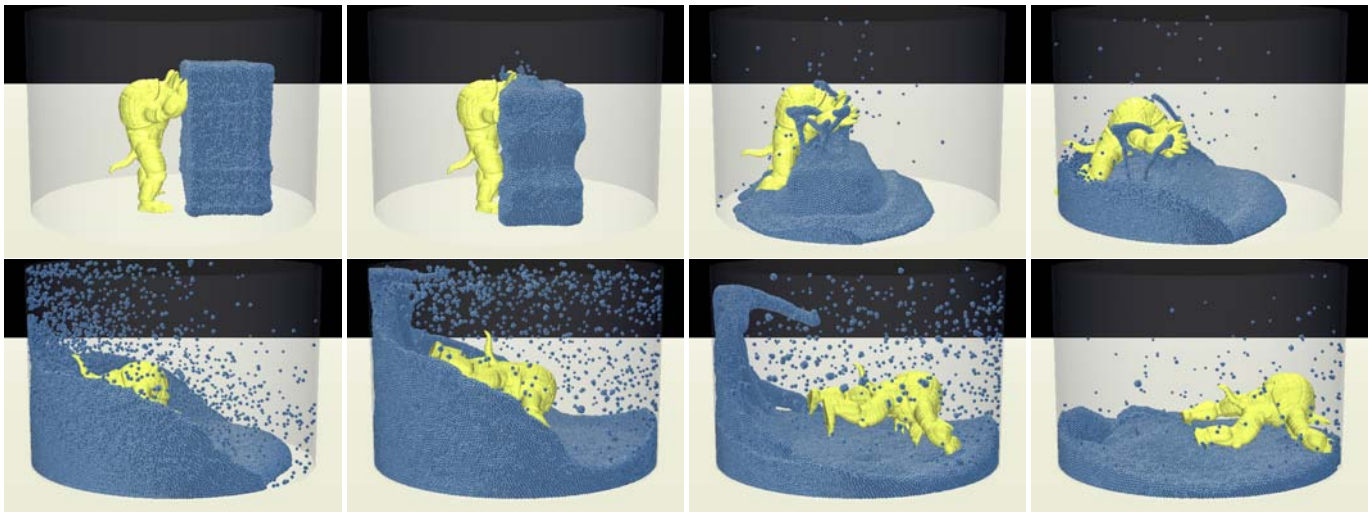


Figure 7: A large number of collisions between fluid and the complicated rigid body. To illustrate the contact points clearly, the fluid particles are rendered as spheres by using pointsprite.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proc. ACM SIGGRAPH 2001*, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing*, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Proc. Computer Graphics International '97*, 178–188.

GÉNEVAUX, O., HABIBI, A., AND MICHEL DISCHLER, J. 2003. Simulating fluid-solid interaction. In *Proc. Graphics Interface 2003*, 31–38.

GINGOLD, R., AND MONAGHAN, J. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181*, 275–389.

HARADA, T., KOSHIZUKA, S., AND KAWAGUCHI, Y. 2007. Smoothed particle hydrodynamics on GPUs. In *Proc. of Computer Graphics International*, 63–70.

HORN, D. R., SUGERMAN, J., HOUSTON, M., AND HANRAHAN, P. 2007. Interactive k-d tree gpu raytracing. In *Proc. Symposium on Interactive 3D graphics and games 2007 (I3D '07)*, 167–174.

KOSHIZUKA, S., AND OKA, Y. 1996. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear Science and Engineering 123*, 421–434.

LEMKE, C. E. 1965. Bimatrix equilibrium points and mathematical programming. *Management Science 11*, 681–689.

LUCY, L. B. 1977. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82*, 1013–1024.

MACMILLAN, W. D. 1960. *Dynamics of Rigid Bodies*.

MIRTICH, B., AND CANNY, J. F. 1995. Impulse-based simulation of rigid bodies. In *Proc. Symposium on Interactive 3D Graphics '95*, 181–188.

MIRTICH, B. 2000. Timewarp rigid body simulation. In *SIGGRAPH*, 193–200.

MONAGHAN, J. 1992. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 543–574.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation*, 154–159.

MÜLLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds 15*, 3-4, 159–171.

PREMOŽE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. T. 2003. Particle-based simulation of fluids. *Computer Graphics Forum 22*, 401–410.

SOLENTHALER, B., SCHLÄFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid–solid interactions: Research articles. *Computer Animation and Virtual Worlds 18*, 1, 69–82.

TAKAHASHI, T., UEKI, H., KUNIMATSU, A., AND FUJII, H. 2002. The simulation of fluid-rigid body interaction. In *SIGGRAPH '02: Sketches & Applications*, 266.

TANAKA, M., SAKAI, M., AND KOSHIZUKA, S. 2007. Particle-based rigid body simulation and coupling with fluid simulation. *Transaction of JSCES*, Paper No.20070007.

YUKSEL, C., HOUSE, D. H., AND KEYSER, J. 2007. Wave particles. *ACM Transactions on Graphics 26*, 3, 99.