Experimenting with Real Time Simulation Parameters for Fluid Model of Soft Bodies

Jaruwan Mesit and Ratan K. Guha
University of Central Florida
4000 Central Florida Blvd. Orlando, Florida, 32816
{jmesit, guha}@eecs.ucf.edu

Abstract

In soft body simulation with fluid modeling, smooth particle hydrodynamics (SPH) is one of the most efficient methods to simulate the soft body for real time applications. In this paper, we introduce a general model of soft bodies with SPH fluid modeling as one of the components for interaction among particles. The fluid force in SPH depends on the density of neighboring fluid particles in the kernel of the considered particle. The fluid force is related to fluid attributes such as fluid density, fluid pressure, and fluid viscosity. Computation becomes faster if the neighboring fluid particles are known during the computations of the fluid attributes. In our simulation of soft body model, the kernels of the fluid attributes are identical, and hence we use the same neighboring fluid particles to evaluate the fluid attributes. In this paper we introduce partitioning and hashing schemes to identify the neighboring fluid particles for SPH to compute the fluid force in the soft body simulation. The suitable parameters for the partitioning and hashing schemes are presented for the modeling. Experimental results show that the grid based scheme can reduce time computation in SPH for fluid modeling in real time applications. We also present a result of a soft body in which the model includes all forces.

Keywords: soft body, smooth particle hydrodynamics

1. INTRODUCTION

Cloth [1,2], hair [3,4,5], elastics [6,7,8], and liquids [9,10] in movie, television, and video game can be simulated as soft body models. Normally a set of points is defined for soft body and, unlike rigid body models, each point moves independently. Individual points in a soft body model have attributes such as position, velocity, and force. Due to the rearrangement of points in soft body to new position in every animation frame, soft body simulation is significantly more complex than rigid body simulation.

In general, the soft body can be simulated by interactions between an arbitrary set of points. Marching Cubes [13] is one of the well known methods to determine a physical surface mesh by surface reconstruction. Such methods are commonly employed to simulate viscous materials like liquids. Other soft body methods simulate semi-soft objects by deforming the points of an existing rigid-body surface without need to periodically recompute the surface mesh. Materials that are simulated by such a method change shape, yet maintain a relatively coherent structure, include cloth, hair , and elastics.

We propose the soft body model, suitable for any application that deforms existing 3D rigid body models. Several parameters, such as mass-spring force, fluid force, and volume pressure force, are applied to a rigid body model to obtain a desired level of deformation. Forces are applied to the mesh vertices, which are connected by a mass-spring system, enabling the model to deform while maintaining a relative configuration among the surface points. Fluid equations and gravitational force create a realistic fluid-like motion on the soft body surface. Volume inside the 3D soft body model is modeled by simulated molecules and ideal gas approximation to maintain the approximated volume of the soft body for real time simulation.

Our soft body model is defined by an initial rigid body state in which the surface points are defined by a hand-modeled triangle mesh. During animation, the various forces deform surface point model away from their pre-defined rigid body state in the following manner.

First, spring and damping forces at the surface push and pull neighboring surface points. Second, ideal gas approximation equations exert pressure force inside the soft body to create volume. Third, gravitational force is applied to all points. Finally, fluid forces induce complex, fluid-like motion at soft body surface. The combination of all forces is applied to each point, positions are updated, and points are collided, resulting in the final deformed shape of the soft body model. Suitable applications for the proposed method include real-time gaming, non-real time graphics, and medical simulations. Forces can be adjusted by parameters to generate different deformations suited for specific applications such as fluid-like soft bodies, human tissue simulation, and blob-like video game monsters.

In this paper, we consider an efficient implementation of fluid forces by SPH modeling. All other forces can be implemented directly by the structures of the model, whereas SPH modeling requires neighboring fluid points in the kernel interacting and generating fluid forces. We propose the partitioning and hashing schemes for efficiently evaluating the fluid forces. The partitioning and hashing schemes use spatial hash function to find the locations of the fluid point and the neighboring fluid points in the kernel. Then fluid force is calculated from the attributes, such as positions or velocities, between the neighboring fluid points and the fluid point. Here, we experiment with parametric values for the partitioning and hashing schemes for SPH fluid modeling. The experiment are set and run in common hardware to compare between with and without the partitioning and hashing schemes. The experiments show that the partitioning and hashing schemes can efficiently be applied to SPH in fluid modeling so that our soft body simulation can be implemented for real time application.

The paper proceeds as follows. Section 2 details the proposed model for fluid-based soft body animation. Section 3 provides the summary of the soft body simulation. Section Sections 4 discuss specifics of the proposed fluid modeling algorithm. Section 5 presents simulation experiments and comparison results. Finally, section 6 concludes the paper and discusses potential further work.

2. MODELING OF SOFT BODIES

In the paper, soft body model with variety of forces is proposed. The soft body 3D model is defined only by surface points where the volume inside the soft body is maintained by simulated gaseous approximation. The model starts with a triangular mesh description of surface points of the 3D model. Mass-spring forces, fluid forces, and gravitational forces are applied to each surface point to model the soft body. Instead of considering all points inside the soft body, we model the inside of 3D soft body by internal pressure force to maintain the approximated volume without internal structure. This approach reduces the computational time and to determine the deformed configuration of the soft body. In addition, we also present partitioning and hashing schemes to minimize computation time of collision detection of the soft body with the environments. Due to the proposed model and the dynamic scheme for collision detection, the simulation is executed in real time.

In general, there are two main ways to deform a triangle mesh: (1) the internal pressure soft body method in [14] and (2) the fluid-based soft body method in [15,16]. Each method provides both benefits and drawbacks.

The internal pressure model with a mass spring system can generate smooth surfaces, which are suitable for balloon-like models. However, a drawback is that the wavy surfaces of liquid-like substances, such as silicone gel, cannot be simulated by this method.

To obtain the benefits of both models while avoiding the drawbacks, the method proposed in this paper utilizes a hybrid model that (1) applies internal pressure force to a mass-spring system surface, and (2) applies fluid equations to more realistically deform the soft body surface. Thus, rippling fluid-like surfaces can be simulated while maintaining the fast rendering speeds of a pre-defined surface mesh.

In this section we formally describe the model of 3D soft body in motion. We first present formal model of a 3D rigid body in motion, $R3D$, as

$$R3D = \langle P, E, T, F, G, V \rangle,$$

where $P = \{p_i \mid i = 0,..,n-1\}$ is a set of surface points of the rigid body and (x_i, y_i, z_i) is a coordinate of surface point p_i,

$E = \{p_i p_j \mid p_i, p_j \in P\}$ is a set of edges that connect p_i and p_j,

$$T = \left\{ \begin{array}{l} p_i p_j p_k \mid p_i, p_j, p_k \in P \text{, and} \\ \mid p_i p_j, p_i p_k, \text{ and } p_j p_k \in E \end{array} \right\}$$ is a set of triangles of rigid body surface defined by p_i, p_j, and p_k,

F is the force at the center of the rigid body,
G is the gravitational force, and
V is the velocity at the center of the rigid body.

In the 3D rigid body in motion, one force and one velocity are applied to the entire rigid body. The edge that connects p_i and p_j is static, as the shape of the body doesn't change during the simulation, in the result that the area of each triangle connected by the edges, $p_i p_j$, $p_i p_k$, and $p_j p_k$, is similarly static. Gravitational force is considered in the rigid model for free fall motion and then force and velocity are calculated and applied to all surface points of the rigid body.

Correspondingly, we formally define the 3D soft body in motion, $S3D$, as

$$S3D = \langle P, E, T, F, N, V \rangle,$$

where P, E, and T are defined in $R3D$,

$F = \{\boldsymbol{F}_i | i = 0,..,n-1\}$ is a set of forces for each surface point of the soft body,

N is proportional to the number of molecules inside the soft body to maintain the approximated volume of the soft body, and

$V = \{\boldsymbol{v}_i | i = 0,..,n-1\}$ is a set of velocities for each surface point of the soft body.

In the 3D rigid or soft body in motion, we also define $l_{ij} = |p_i p_j|$, length of the edge $p_i p_j \in E$ and $a_{ijk} = |p_i p_j p_k|$, area of the triangle, $p_i p_j p_k \in T$. l_{ij} and a_{ijk} are constants for rigid body whereas l_{ij} and a_{ijk} change in soft body due to deformation and interaction with environment. In this model, it is assumed that the number of surface points, edges, and triangles remain constant after any deformation.

In this soft model, surface elasticity is created by a mass-spring system in which surface mesh points move freely, yet retain a relative configuration to the original mesh. For mass-spring force, $\boldsymbol{F}^t_{mi}$ is determined by two parts, spring force, $\boldsymbol{F}^t_{si}$, and damping force, $\boldsymbol{F}^t_{di}$. Thus, $\boldsymbol{F}^t_{mi}$ is defined as follows [17,18]:

$$\boldsymbol{F}^t_{mi} = \boldsymbol{F}^t_{si} + \boldsymbol{F}^t_{di}, \tag{2.1}$$

where $\boldsymbol{F}^t_{si}$ is spring force at surface point i and $\boldsymbol{F}^t_{di}$ is damping force at surface point i.

In order to model volume of the soft body, an internal pressure, $\boldsymbol{F}^t_{pi}$, must push the surface points outward. This volume is created by pressure force generated by the molecules within the soft body [14].

In the real-world gravity affects all objects, thus the simulation model must account for it. The force of gravity, $\boldsymbol{F}^t_{gi}$, experienced by an object earth is equal to weight of the object experiencing the gravitational pull.

For modeling behavior of fluids on the surface of soft body, we consider the impact of fluid pressure and fluid viscosity of the fluid. For that purpose, we simulate the surface points as free moving particle interacting with nearby surface points within a radius. Since the mass-spring system provides surface tension, we consider both fluid pressure force generated due to the density and fluid viscosity force generated due to the viscosity of the fluid. To model these forces, we adopt the smoothed particle hydrodynamics (SPH) model developed originally for astrophysical problem and later used in interactive application of particles based on fluid simulation [15,16]. SPH is an interpolation method for particle system. SPH distributes quantities in a local neighborhood of each particle using radial symmetrical smoothing kernels. We utilize poly6, spiky, and viscosity smoothing kernels in [15,16] to model fluid density, fluid pressure, and viscosity forces. The smoothing kernel, $W(r, h_w)$, is a function for the smoothing kernel with the radius, h_w, and the distance, r, where r is calculated from the positions of surface points i and j. The gradient and laplacian of smoothing kernel are presented as $\nabla W(r,h)$ and $\nabla^2 W(r,h)$, respectively.

Thus, during fluid force calculation, fluid density and fluid pressure are computed to generate fluid pressure force and fluid viscosity force as presented as follows.

Following the model developed in [15,16], fluid density is given by:

$$\rho^t_i = \sum_j m_j W_{poly6}(l^t_{ij}, h_w), \quad \forall j \text{ such that } l^t_{ij} \le h_w, \tag{2.2}$$

where ρ^t_i is the density at surface point i at time t, m_j is the mass at surface point j, and h_w is the core radius of SPH.

Next, fluid pressure is generated from fluid density as:

$$P^t_i = k(\rho^t_i - \rho_0), \tag{2.3}$$

where P^t_i is the pressure at surface point i at time t, k is the gas constant, ρ^t_i is the density at surface point i at time t, and ρ_0 is the initial density.

Fluid pressure force at the soft body surface point i, $\boldsymbol{F}^t_{fpi}$, is computed as:

$$\boldsymbol{F}^t_{fpi} = -\sum_j m_j \frac{P^t_i - P^t_j}{2\rho^t_j} \nabla W_{spiky}(l^t_{ij}, h_w), \quad \forall j \text{ such that } l^t_{ij} \le h_w, \tag{2.4}$$

where m_j is the mass at surface point j, P_i^t and P_j^t are pressure values at surface points i and j respectively at time t, ρ_j^t is the density at surface point j at time t, and h_w is the core radius of SPH.

Finally the fluid viscosity force at the soft body surface point i, $\boldsymbol{F}_{fvi}^t$, is generated by:

$$\boldsymbol{F}_{fvi}^t = \mu \sum_j m_j \frac{\boldsymbol{v}_j^t - \boldsymbol{v}_i^t}{\rho_j^t} \nabla^2 W_{viscosity}(l_{ij}^t, h_w), \ \forall j \text{ such that } l_{ij}^t \le h_w, \tag{2.5}$$

where μ is the viscosity of fluid, m_j is the mass at surface point j, $\boldsymbol{v}_i^t$ and $\boldsymbol{v}_j^t$ are the velocities at surface points i and j respectively at time t, ρ_j^t is the density at surface point j at time t, and h_w is the core radius of SPH.

The fluid force, $\boldsymbol{F}_{fi}^t$, is the combination of two different forces; (1) fluid pressure force, $\boldsymbol{F}_{fpi}^t$, and (2) fluid viscosity force, $\boldsymbol{F}_{fvi}^t$. Hence, we define the fluid force as follows:

$$\boldsymbol{F}_{fi}^t = \boldsymbol{F}_{fpi}^t + \boldsymbol{F}_{fvi}^t \tag{2.6}$$

where $\boldsymbol{F}_{fpi}^t$ is fluid pressure force at surface point i and $\boldsymbol{F}_{fvi}^t$ is fluid viscosity force at surface point i.

When we compute all individual forces on a soft body, the forces must be combined in this step for each surface point. The combination of all forces proportionally selects of particular forces suitable for specific applications.

To avoid stability issues, we use the implicit method in [19] to update soft body motion. In our model, velocity is generated from combination of all force. Thus, the velocity at surface point i at time $t+h$, $\boldsymbol{v}_i^{t+h}$, is calculated by:

$$\boldsymbol{v}_i^{t+h} = \boldsymbol{v}_i^t + \Delta \boldsymbol{v}_i^{t+h}, \tag{2.7}$$

$$\boldsymbol{v}_i^{t+h} = \boldsymbol{v}_i^t + \boldsymbol{F}_i^{t+h} \frac{h}{m_i}, \tag{2.8}$$

To compute the force at surface point i at time $t+h$, $\boldsymbol{F}_i^{t+h}$, we use the implicit method that presented in [19] where $\boldsymbol{F}_i^{t+h}$ is derived from the force at surface point i at time t, $\boldsymbol{F}_i^t$. Then the surface point i at time $t+h$, p_i^{t+h} is as follows:

$$p_i^{t+h} = p_i^t + \boldsymbol{v}_i^{t+h} h, \tag{2.10}$$

where p_i^{t+h} and $\boldsymbol{v}_i^{t+h}$ is the position and velocity of surface point i at time $t+h$, p_i^t is the position of surface point i at time t, and h is the time interval between simulation steps.

3. SIMULATION OF SOFT BODY MODEL

In the simulation of soft body model, the spring force and damping force are simply calculated by the lengths and velocities from the connecting points. We have defined an array for this structure at the beginning of our simulation. In internal pressure force, the areas of the triangles on the surface of model is involved which is determine by the structure of the model. This pressure force is easily computed by using the model structure for the area where the internal pressure force is acting on. Similarly, the gravitational force is merely exerted into each point by the relation of the surface point mass and the gravity. On another hand, the fluid force depends on the free moving particles and the neighboring particles. The particles change the positions in every animated frame, so the efficient method is required to determine neighboring particles. We propose partitioning with hashing techniques to determine the neighboring surface points. Thus, the details neighboring surface points by hashing for fluid model is explained in the next section.

4. SIMULATION OF FLUID MODELING

For fluid attributes of fluid force such as fluid density, fluid pressure force, and fluid viscosity, these attributes are based on the neighboring surface points in the kernel or radius of the considering surface point. Each attribute of fluid force at one particular surface point requires the knowledge of the attributes of neighboring surface points within the radius of the surface point. The ordinary implementation for fluid force is point-by-point method where all the point has to be tested if they are in the kernel or radius of the considering point or not. Since there are three attributes for fluid force thus the testing to find the neighboring surface points within the radius of the point has to perform three times from the first surface point to the last surface point. In this paper we present the partitioning and hashing schemes to reduce the time computation of finding the neighboring surface point for each surface point in fluid modeling. The detail of the partitioning and hashing schemes to find neighboring surface point is described here.

4.1 Neighboring Surface Points by Hashing for Fluid Model

In order to find neighboring surface points, all surface point distances must be compared, which is On^2 in time complexity. To reduce the time complexity, we propose a custom spatial hashing to partition 3D space into grid cells. During the animation, the grid cell of each surface point in the frame is determined by spatial partitioning. The surface point within that grid cell is then mapped into 3D hash table. Since in eq. 2.2, 2.4, and 2.5 in fluid modeling concern only the surface points in the SPH core radius, the surface points outside the radius can be ignored. The determination of neighboring points of a surface point is accomplished in two steps: (1) creating a 3D hash table for all surface points, and (2) obtaining a list of neighboring points for each surface point using the 3D hash table.

4.1.1 3D Hash Table Creation

To determine the neighboring surface points of a particular surface point, we partition the environment in the simulation into cubic grid cells. Each grid cell has the same size as the kernel of fluid model represented by SPH core diameter, $2h_w$, where h_w is core radius of SPH. In this partitioning, a three dimensional (3D) array for these grid cells requires a large amount of memory space. To avoid large memory allocation for the large number of grid cells to search for the neighboring surface points, we propose a set of lists indexed by a 3D hashing scheme to utilize memory space effectively and develop an efficient search technique for neighboring surface points. In this hashing scheme, we map the surface points into a small 3D hash table. Since the hash table size is an important parameter for the execution time of each animated frame, we have experimented with different 3D hash table sizes (prime numbers) containing the surface points ranging from 500 to 6000 points. The results of our experiments are presented in section 5

Conceptually, our spatial partitioning method divides the environment into grid cells. However in reality we do not create all these grid cells. Instead, for each surface point we determine the grid cell index, in which the surface point belongs. This grid cell index of the surface point is mapped into the hash table index. The hash table then places that surface point into the list pointed by the hash index shown in figure 1. The grid cell index and hash table index of the surface point p are as follows:

The surface point $p = (x, y, z)$ belongs to the grid cell $c = (c_x, c_y, c_z)$, where $c_x = \lfloor x / 2h_w \rfloor$, $c_y = \lfloor y / 2h_w \rfloor$, and $c_z = \lfloor z / 2h_w \rfloor$, and h_w is the core radius of SPH. (4.1)

The hash index $H = (H_x, H_y, H_z)$ of the grid cell $c = (c_x, c_y, c_z)$ is determined by

$$H_x = c_x \textbf{ mod } HT \quad , \quad H_y = c_y \textbf{ mod } HT \quad , \text{ and } H_z = c_z \textbf{ mod } HT \, , \tag{4.2}$$

where HT is the hash table size in each direction. Hence, HT^3 is the total size of the hash table.

Let's us assume that the minimum and maximum grid cell index in x , y , and z directions are given by $(minc_x, maxc_x)$, $(minc_y, maxc_y)$, and $(minc_z, maxc_z)$.

The neighboring grid cells $c' = (c'_x, c'_y, c'_z)$ of the grid cell $c = (c_x, c_y, c_z)$ is given by

$$c'_x = c_x + i \, , \; c'_y = c_y + i \, , \text{ and } c'_z = c_z + i \, , \tag{4.3}$$

where $i = 0$, ± 1 and $minc_x \le c'_x \le maxc_x$, $minc_y \le c'_y \le maxc_y$, $minc_z \le c'_z \le maxc_z$.

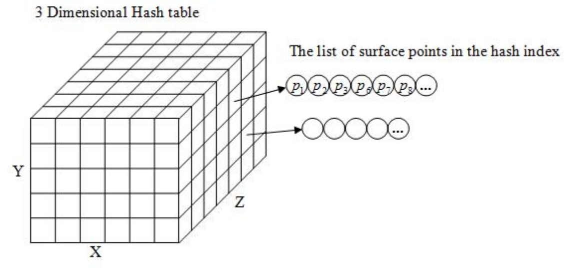

3 Dimensional Hash table
The list of surface points in the hash index
p_1 p_2 p_3 p_6 p_7 p_8 ...
...
Y
X
Z

Figure 1. 3D hash table for SPH Each grid cell index $c = (c_x, c_y, c_z)$ is mapped into the hash index $H = (H_x, H_y, H_z)$.

Then the hash index $H' = (H'_x, H'_y, H'_z)$ of a neighboring cell $c' = (c'_x, c'_y, c'_z)$ is

$$H'_x = c'_x \textbf{ mod } HT \quad , \quad H'_y = c'_y \textbf{ mod } HT \quad , \text{ and } H'_z = c'_z \textbf{ mod } HT \; . \qquad (4.4)$$

4.1.2 Obtaining List of Neighboring Points from The 3D Hash Table

For the surface point p, fluid force in SPH method is affected by the surface points within the core radius h_w. We create the table called, *fluid particle list*, to contain neighboring surface points of surface point p, shown in figure 2. Each element p of the fluid particle list table has the pointer that points to the surface points within the core radius h_w of the surface point p.

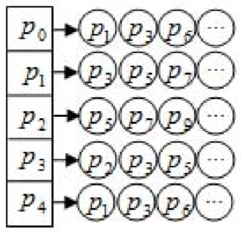

p_0 → p_1 p_3 p_5 ···
p_1 → p_3 p_5 p_7 ···
p_2 → p_5 p_7 p_9 ···
p_3 → p_2 p_3 p_5 ···
p_4 → p_1 p_3 p_6 ···

Figure 2. **Fluid particle list table** Each soft body object has a fluid particle table of size n, where n is the number of surface points in the soft body. Each entry in the table lists the surface points within the kernel radius of the surface point correlating to that table entry.

Each surface point p has the grid cell index $c = (c_x, c_y, c_z)$ and the hash index $H = (H_x, H_y, H_z)$ indicated in eq. 4.1 and 4.2. All surface points in the hash index $H = (H_x, H_y, H_z)$ is tested to determine if they are within the core radius, h_w, of the surface point p. If so, the surface points within the radius of the surface point p are placed into the fluid particle list table at the index for surface point p as shown in figure 3. Since the surface points within the core radius, h_w, of the surface point p may reside in the neighboring grid cells $c' = (c'_x, c'_y, c'_z)$ with the hash indices $H' = (H'_x, H'_y, H'_z)$ indicated in eq. 4.3 and 4.4, the list of surface points in the hash indices $H' = (H'_x, H'_y, H'_z)$ are also checked for the surface points within the core radius, h_w, of surface point p.

For each surface point, we have determined a list of neighboring surface points. These neighboring surface points affect the fluid density, fluid pressure force, and fluid viscosity force at surface point p in fluid modeling. After fluid modeling, the fluid force is combined with other forces as detailed in Section 3. Motion physics are then applied to move particles through space. In the next section the experiment of hash table size for fluid modeling is investigated.

5. EXPERIMENTS FOR HASH TABLE SIZE FOR FLUID MODELING

The goal of the experiments in this section is first to find the optimal hash table size that gives the best results depending on the number of particles in the simulation. Second, comparison between the point-by-point method and partitioning and hashing schemes are drawn. Then, performance of the partitioning and hashing schemes is presented for varying numbers of particles. Finally, the performance is measured when the numbers of particles that we use in our simulation are applied. All benchmarks presented are conducted on a mid-performance laptop, a 2.0 GHz Core-2 Dui processor, with 1Gb RAM, and an NVIDIA GeForce 6800 Mobile GPU.

5.1. Hash Table Size

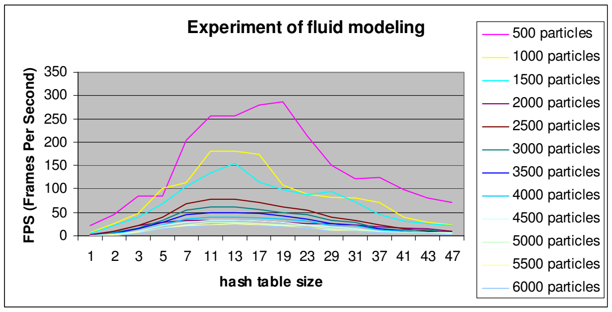
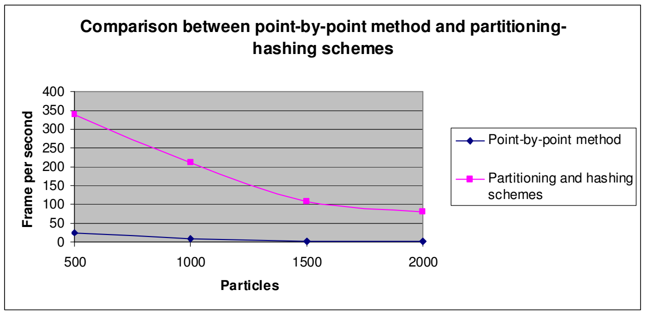


Figure 3. The experiment of hash table size for fluid modeling The hash table size of 1^3, 2^3, 3^3 and 47^3 are examined. The best result depends on the number of particles. The tend shows that the better result starting from hash table size of 13^3 and getting worse from hash table size of 23^3 depending on the number of particles.

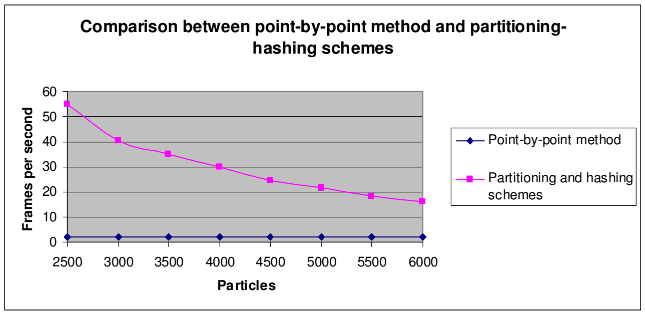
In this experiment the number of fluid particles is varied from 500 to 6000. The hash table size is changed for each experiment and the performance of the partitioning and hashing schemes is measured in frames per second (FPS). In this experiment a three dimensional hash array for *x*, *y*, and *z* dimensions is utilized. Each dimension of array is a prime number in the set 1, 2, 3, …, and 47 which means the sizes of the hash table are 1^3, 2^3, 3^3 and 47^3. Results displayed in figure 3 show that the best performance is obtained when the array size is 13^3 in general.

5.2. Point-by-point Method Compared to Partitioning and Hashing Schemes

Since the best result from the previous experiment at the hash table size of 13^3, this experiment compares point-by-point method and the partitioning and hashing schemes. The number of particles is varied from 500 to 6,000 and performances of the methods are measured in average FPS. With point-by-point method, the simulation can run at 10 FPS on average and drop to 2 FPS when the number of particles increases to 6000 particles in the simulation. As shown in figure 4, with partitioning and hashing schemes and array size of 13, the simulation runs at 338 FPS for 500 particles and still runs at 16 FPS for 6,000 particles.



(a)



(b)

Figure 4. The comparison of point-by-point method and partitioning and hashing schemes with array size of 13. The experiment is set by using the hash table size of 13^3 and the numbers of particles are from 500 to 6000.

5.3. Performance of Partitioning and Hashing Schemes for Different Numbers of Particles and Frame Number

This experiment measures the performance of the partitioning and hashing schemes where the number of particles is varied between 500 and 6,000 and the hash table size is set to 13^3. The results are captured from the frame 100th to 3000th. On average, the results presented in figure 5 show that the partitioning and hashing schemes provide 343 FPS for 500 particles to 16 FPS for 6000 particles and the simulation remains stable.

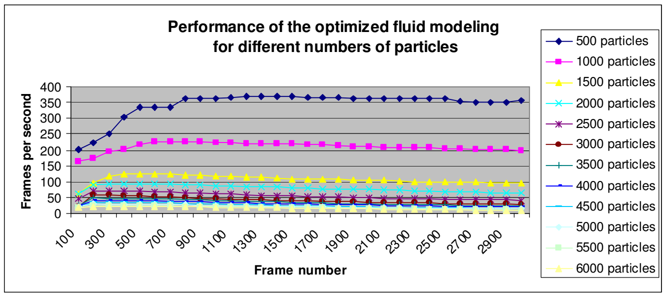


Figure 5. Performance of the partitioning and hashing schemes The detail of time computation is presented in every one hundred frames. In this experiment the simulation starts to become stable at frame 1000th.

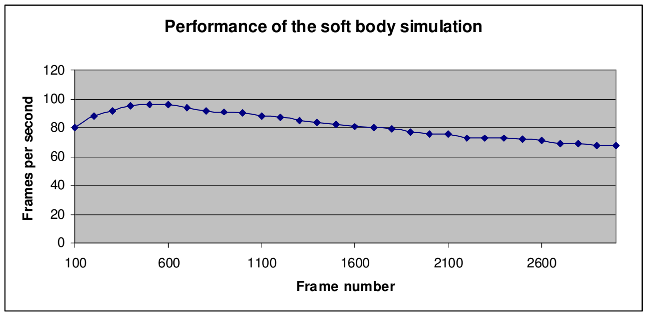


Figure 6. Performance of soft body simulation Since the soft bodies of our simulation are composed of 1,986, this number of particles is examined in this experiment to present whether the real time application can be simulated with the proposed implementation. This figure shows that the simulation can run in real time.

5.4. Fluid Modeling Performance by Partitioning and Hashing Schemes

In this experiment, the partitioning and hashing schemes are tested with particles counts of 1986. The graph in figure 6 shows the performance of the method by in FPS over 3000 frames of animation, demonstrating this numbers of particles can be simulated in real time. The lowest FPS in the experiments is 16, which is nearly acceptable for real time, and would be substantially higher better hardware (note the experiments are performed on a mid-range laptop).

5.5 Simulation of Soft Bodies

The result of soft body simulation is shown in figure 7. We apply the fluid modeling on the surface of the soft body. The technique of the partitioning with hashing schemes is applied on the surface points of the soft body. There are 1,986 surface points, 3,968 faces, and 5,952 springs on the soft body. The simulation results to display more than 20 FPS.

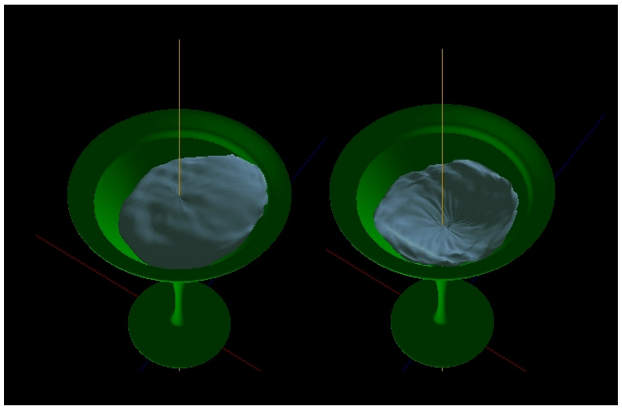


Figure 7. **Simulation of soft bodies** The figure shows the result of soft body simulation where the fluid modeling is applied on the surface of the soft bodies.

6. CONCLUSION

In our soft body simulation, the soft body 3D model is defined only by surface points. The model starts with a triangular mesh description of surface points of the 3D model. Mass-spring force, fluid force, and gravitational force are applied to each surface point to model the soft body. Instead of considering all points inside the soft body, we model the inside of 3D soft body by internal pressure force to maintain the approximated volume without internal structure. This approach reduces the computational time and to determine the deformed configuration of the soft body. In our implementation of fluid modeling, we present the partitioning and hashing schemes to reduce time computation in fluid modeling. The experiments in this paper indicate that our implementation scheme can sufficiently reduce computation time for SPH in fluid modeling so that the simulation can run in real time. For future works more complex models can be considered for different applications and efficient collision detection between objects should be implemented in the simulation.

Reference :

[1] Baraff, D. and Witkin, A. 1998. "Large steps in cloth simulation," *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, New York, New York, USA: ACM Press, 43-54.

[2] Kang, Y., Choi, J., and Cho, H. 2000. "Fast and stable animation of cloth with an approximated implicit method," *Computer Graphics International*.

[3] Ward, K., Galoppo, N., and Lin, M. 2006. "A Simulation-based VR System for Interactive Hairstyling," *IEEE Virtual Reality Conference (VR 2006)*, 257-260.

[4] Ward, K., Bertails, F., Kim, T., Marschner, S.R., Cani, M., and Lin, M.C. "A survey on hair modeling: styling, simulation, and rendering.," *IEEE transactions on visualization and computer graphics*, vol. 13, 213-34.

[5] Ward, K., Galoppo, N., and Lin, M., 2007. "Interactive Virtual Hair Salon," *Presence: Teleoperators & Virtual Environments*, 237-251.

[6] Terzopoulos, D., Platt, J., and Barr, A., 1987. "Elastically deformable models," 205-214.

[7] Costa, I.F. and Balaniuk, R., 2001. "LEM-an approach for real time physically based soft tissue simulation," 2337-2343.

[8] Cotin, S., Delingette, H., and Ayache, N., 1999. "Real-time elastic deformations of soft tissues for surgery simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, 62-73.

[9] Foster, N. and Fedkiw, R., 2001. "Practical animation of liquids,"

[10] Hinsinger, D., Neyret, F., and Cani, M.P., 2002. "Interactive animation of ocean waves,"

[11] Kim, J., Cha, D., Chang, B., Koo, B., and Ihm, I., 2006. "Practical animation of turbulent splashing water,".

[12] Losasso, F., Shinar, T., Selle, A., and Fedkiw, R., 2006. "Multiple interacting liquids," *ACM Transactions on Graphics*, vol. 25.

[13] Lorensen, W.E. and Cline, H.E., 1987. "Marching cubes: A high resolution 3d surface construction algorithm," .

[14] Matyka, M. and Ollila, M., 2003. "Pressure Model of Soft Body Simulation,".

[15] Müller, M., Charypar, D., and Gross, M., 2003. "Particle-based Fluid simulation for interactive applications,".

[16] Müller, M., Solenthaler, B., Keiser, R., and Gross, M., 2005. "Particle-Based Fluid-Fluid Interaction,".

[17] Bourg, D.M., *Physics for game developers*, O'reilly,

[18] Desbrun, M., Schröder, P., and Barr, A., 1991. "Interactive animation of structured deformable objects,".

[19] Mesit, J., Chaudhry, S., and Guha, R., 2007. "3D Soft Body Simulation Using Mass-spring System with Internal Pressure Force and Simplified Implicit Integration," *JOURNAL OF COMPUTERS*, vol. 2.