

Space-Time Adaptive Simulation of Highly Deformable Substances

Mathieu Desbrun, Marie-Paule Cani

► To cite this version:

Mathieu Desbrun, Marie-Paule Cani. Space-Time Adaptive Simulation of Highly Deformable Substances. [Research Report] RR-3829, INRIA. 1999. inria-00072829

HAL Id: inria-00072829

<https://hal.inria.fr/inria-00072829>

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Space-Time Adaptive Simulation of Highly Deformable Substances

Mathieu Desbrun Marie-Paule Cani
Caltech iMAGIS

mathieu@cs.caltech.edu Marie-Paule.Cani@imag.fr

No 3829

Décembre 1999

_____ THÈME 3b _____



*apport
de recherche*

Space-Time Adaptive Simulation of Highly Deformable Substances

Mathieu Desbrun Marie-Paule Cani
Caltech iMAGIS
mathieu@cs.caltech.edu Marie-Paule.Cani@imag.fr

Thème 3b — Interaction homme-machine,
images, données, connaissances
Projet iMAGIS

Rapport de recherche n° 3829 — Décembre 1999 — 22 pages

Abstract: This report presents an approach for efficiently yet precisely simulating highly deformable substances ranging from solids to liquids. The key idea is to use a *state equation* for specifying the dynamics of the substance. During a simulation, the material is sampled by particles that derive their interaction forces from this state equation. Since this ensures the same qualitative behavior whatever the discretization rate, an adaptive scheme can be used during simulations: the particle system adapts over space and time according to a given compromise between precision and computational efficiency. The system refines (i.e., particles are subdivided) in areas undergoing large or fast deformations, while it simplifies (i.e., neighboring particles are merged) in stable regions. Meanwhile, the values of the individual integration time steps used for each particle are automatically adapted to avoid instabilities. An active implicit surface is used to visualize the substance. It smoothly coats the particles and filters over time internal changes of granularity.

Contact: mathieu@cs.caltech.edu, Marie-Paule.Cani@imag.fr

Key-words: Simulation, deformable objects, particle system, space-time adaption

(Résumé : *tsvp*)

Simulation adaptative en temps et espace pour la simulation de matériaux très déformables

Résumé : Ce rapport de recherche présente une nouvelle approche de simulation rapide et prédictive de matériaux très déformables, solides ou liquides. L'idée directrice est d'utiliser une *équation d'état* pour spécifier le comportement macroscopique d'une substance. Durant la simulation, le matériau est échantillonné par des particules dérivant leurs forces d'interaction de l'équation d'état. Le même comportement qualitatif étant assuré quelque soit la finesse de discrétisation, un schéma adaptatif peut être utilisé: le système particulaire s'adapte en temps et en espace, afin de fournir un bon compromis entre efficacité et précision des calculs. Les particules se subdivisent donc dans les régions où de fortes déformations ont lieu, alors qu'elles se regroupent dans les régions stables pour minimiser les calculs. A tout instant, le pas de temps d'intégration est automatiquement déterminé pour assurer une stabilité numérique adéquate. Enfin, une surface implicite active englobe le tout, permettant une visualisation efficace de la surface du matériau tout en masquant les changements de granularité du modèle interne.

Mots-clé : Simulation, objets déformables, système de particules, adaptation en temps et espace.

1 Introduction

The problem of animating highly deformable substances has captured a lot of attention up to now, despite of the difficulty to simulate and to render them realistically. Ranging from the simulation of gaseous phenomena [26, 18] to the animation of viscous substances [20, 28, 7], most models rely on particle systems animated under simplified laws of physics. Such models dispense the animator from the laborious design of key images. However, finding parameter values of a physically-based system in order to produce a specific behavior is not an easy task. Many trials are usually required to get a fine and stable animation. The difficulty is increased by the fact that modifying the number of particles used in these models or the value of the integration time step may produce unpredictable changes of the simulated behavior.

This paper presents a method for increasing efficiency while easing the animator’s task. Designed for highly deformable substances, our adaptive model describes the desired behavior independently of the space or time resolution. These discretizations are automatically and locally adjusted during simulation, in order to optimize computations while avoiding divergence.

1.1 Related work on highly deformable materials

A first approach to simulate fluid-like materials is through an Eulerian simulation, extensively used in hydrodynamics. A fixed grid stores mass density and velocity of the simulated fluid over time. Very realistic animation of water or gas can be obtained [10, 11] through the visualization of markers evolving in the velocity field. Multigrid schemes could be implemented to improve efficiency. However, hydrodynamics equations are ill-conditioned for stiffer substances such as very viscous liquids, dough, mud, or clay. In addition, Eulerian approaches only handle collisions with obstacles that are aligned with the voxels, which may be limitative.

Physically-based particle systems have been extensively used in Computer Graphics for animating unstructured substances that may separate and re-organize in various ways [20, 27, 19]. Matter is discretized according to a Lagrangian formalism, i.e., into mass elements that interact through long range attraction and short range repulsion forces. These forces, although applied here at a totally different scale, derive from the Lennard-Jones model extracted from microscopic observations [20]. Since a set of punctual particles cannot be rendered directly, implicit surfaces — i.e., iso-surfaces of scalar fields generated by the particles — have been used for visualizing the animations for substances ranging between solids and liquids [28, 7].

In spite of the easiness of implementation, performing complex simulations with a particle system is not an easy task. Firstly, space resolution, characterized by the number and the size of particles, must be chosen beforehand. This *granularity* should not be visually perceived, which usually results in large numbers of particles and thus long running times. The integration time step is also difficult to adjust, being directly responsible for instabilities if too large and for inefficiency if too small. Lastly, the tuning of the simulated behavior through the parameters of interaction forces requires much skill, especially as the results heavily depend on the number and size of particles, as well as on the integration time step. In other words, the major default of conventional particle systems is that the simulated behavior *emerges* from the coupled forces that act between pairs of particles. Spatial resolution is fixed during simulations, since modifying it could produce dramatic changes of the simulated behavior. This results in simulations that, apart for being non intuitive and so difficult to set up, are far from efficient.

1.2 Emergence of Adaptive Models

Adaptive methods arise in many fields of Computer Graphics. They express the key idea of *automatically concentrating computational effort where and when needed*. These methods adaptively refine or simplify a current model in order to ensure accuracy while saving computation.

This general paradigm has already been successfully used in global illumination methods: Radiosity has gained tremendous efficiency and reliability with hierarchical algorithms that automatically subdivide or cluster surface patches to ensure a given accuracy at low cost [12, 25]. In visualization too, adaptive approximation of objects is achieved for instance by mesh simplification in over-sampled areas and mesh refinement near the viewer [14, 4].

Closer to our concern, an adaptive framework based on particles has been developed for the interactive sampling and control of implicit surfaces [29]. A set of repulsive particles constrained to stay onto the iso-surface spread rapidly over under-sampled areas by adapting their radius of repulsion. At the same time, fissions and deletions of particles are used to obtain the desired sampling density at equilibrium.

The level of detail paradigm appeared more recently in animation [3]. Here, the main problem is to adapt models while ensuring that each level of detail still approximates a given and well-defined behavior. An adaptive physically-based model for deformable materials simulation was developed for hanging visco-elastic clothes [15]. The cloth's shape is computed from a mass-spring network that locally refines when the angle between two adjacent springs exceeds a given threshold. For each new mass added, the time step is halved, while stiffness of neighboring springs is doubled (this keeps the deformation wave speed at a constant value). However, the method does not guarantee that the surface keeps a similar underlying physical behavior during the animation. The global mass changes over time, even if collisions are treated properly. Moreover, the problem of simplifying back the network when possible is not addressed.

1.3 Overview

This paper presents an adaptive method for animating viscous substances ranging between solids and liquids, that preserve their volume but may separate or melt during animations. Our approach relies on a deformable model whose deformations are constrained by a state equation [8]. It defines the material with a global property regardless of the spatial or temporal resolution, and thus provides a good framework for an adaptive animation scheme.

During a simulation, the matter is sampled by particles. Attraction/repulsion forces between them are derived from the state equation, thus yielding different approximations of the same behavior whatever the sampling resolution. Local fissions and fusions of particles are automatically generated in order to optimize computation while keeping a prescribed accuracy. As a result, fewer and larger particles are used in stable areas while refinements occur where the substance undergoes deformation. We then adapt the individual time step of each particle to avoid instabilities. Finally, an active surface model based on an implicit formulation smoothly coats the particles, providing an enhanced yet efficient visualization of the animation.

The remainder of this paper develops as follows: Section 2 reviews the deformable model introduced in [8]. Section 3 extends this model to particles of different masses. The problem of adaptively discretizing space and time are respectively addressed in Sections 4 and 5. Section 6 describes visualization with an active implicit surface. Results and performance are presented in Section 7. Section 8 then concludes with future work.

2 A density-based deformable model

This section reviews the density-based deformable model presented in [8]. This animation technique was inspired from *Smoothed Particles Hydrodynamics*, a method initially developed by astrophysicists [21], that has been shown equivalent to a Galerkin finite element method in space and time [9].

2.1 State equation

A macroscopic definition of a deformable model can be provided by a state equation that governs the evolution of a *pressure* field P inside the substance. Conservative internal forces, called “pressure forces”, are proportional to the gradient of pressure [1]. Most often, these forces are combined with dissipative forces that model internal friction inside the deformable body.

In order to animate a viscous substance that comes back to a constant volume when no external force is applied, the state equation can be set to [8]:

$$P = k(\rho - \rho_0) \quad (1)$$

This generates pressure forces that tend to restore a rest density ρ_0 when the current density ρ has become too high or too low. The parameter k in equation (1) controls the strength of density recovering, and is thus analogous to a stiffness parameter.

2.2 Space discretization with smoothed particles

Once the state equation is set, a method for simulating the substance must be provided. Rather than discretizing space with a fixed grid and studying what flows in and out of each voxel over time as in Eulerian approaches, the fluid is sampled by a set of mass elements, called *smoothed particles* [21, 23, 8]. A particle i represents a fixed mass m_i , distributed in space around the current position \mathbf{x}_i of the particle according to a “smoothing kernel” W_h . The smoothing length h controls the extend in space of the mass distribution. The integral of W_h equals to one for all h , and W_h tends to the Dirac delta function when h tends to zero. A possible equation for W_h is given in Appendix B.

Smoothed particles are used as sample points for discretizing continuous physical fields inside the substance, such as mass density and pressure: The values of these fields at particle i are respectively denoted ρ_j and P_j . Field values can then be evaluated from these samples at any point in space through a Monte-Carlo approximation of a convolution (see Appendix A), yielding:

$$f(\mathbf{x}) = \sum_j f_j \frac{m_j}{\rho_j} W_h(\mathbf{x} - \mathbf{x}_j) \quad (2)$$

where the f_j are the sample values of f at particle positions. Smoothed particles also provide an approximation of the field’s gradient :

$$\nabla f(\mathbf{x}) = \sum_j f_j \frac{m_j}{\rho_j} \nabla W_h(\mathbf{x} - \mathbf{x}_j) \quad (3)$$

Applied to pressure, we can derive (see Appendix C for proof) the following symmetric expression for pressure forces, enforcing the action-reaction principle:

$$\mathbf{F}_i^{\nabla P} = \sum_{j \neq i} -m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i W_h^{ij} \quad (4)$$

where $\nabla_i W_h^{ij}$ denotes the gradient of $W_h(\mathbf{x}_i - \mathbf{x}_j)$ taken with respect to the coordinates of particle i . These forces can be interpreted as the sum of pairwise opposite forces exerted on each pair of particles. Pairwise forces may be either repulsive or attractive according to local density values, and vanish when the distance between two particles exceeds $2h$. These attraction/repulsion interaction forces thus naturally appear from the state equation, instead of being arbitrarily fixed.

2.3 Animating the density-based model

To perform a simulation, the volume initially occupied by the substance is first regularly sampled by particles of equal mass m_i . Then, as in any particle system, the equation of motion for each particle is integrated over time. At each time step :

1. Compute new values of mass density at particles positions, according to equation (2) with $f \equiv \rho$.
2. Deduce pressure at particle positions from the state equation (1).
3. Compute pressure forces given by equation (4).
4. Add other forces such as viscosity (see Appendix D) and gravity.
5. Integrate the equation of motion of each particle, i.e., update its acceleration, speed and then position from the set of applied forces.

This density-based model thus results in an implementation similar to conventional particle systems. A fixed grid over space can be used to perform nearest neighbors search, reducing computation to a linear algorithmic cost with respect to the number of particles [20]. Yet the model provides specific advantages:

- Interaction forces between particles are derived from a state equation rather than being arbitrarily designed by the user. The state equation provides a macroscopic description of the material, which is defined through three simple parameters: mass density, stiffness, and viscosity.
- The Monte-Carlo interpretation of SPH ensures that stability increases with the number of particles. Particles can thus be seen as sample points inside the material.
- Stability criteria for the integration time step choice can be derived from the *speed of sound* c of the simulated material, which represents the fastest velocity at which a deformation may propagate. Since pressure waves propagate at speed $c = \sqrt{\partial P / \partial \rho}$ [1], equation (1) induces:

$$c = \sqrt{k} \quad (5)$$

3 Towards Space-Time Adaptive Simulation

As the density-based model does not rely on any hypothesis on spatial or temporal resolution, different approximation levels of the same substance can be obtained by simply changing the number of smoothed particles and the value of the elementary mass m_i . This is a major advantage over conventional particle systems. However, using a large number (several hundreds) of small particles to obtain a finer animation with a smaller time step to avoid divergence leads to unbearable running time.

To alleviate the computational burden, we propose an adaptive framework for the simulation of highly deformable substances. In addition to enhancing both accuracy and efficiency, this framework completely frees the user from the non intuitive specification of space and time resolutions. The basic idea is to adaptively divide particles in areas experiencing high variations of pressure, while merging neighboring ones in stable areas, where pressure is almost uniform. This approach is very close in spirit to the fission/death process defined in [29]. However, since we must enforce physical laws, this is an altogether different problem. For instance, a mere deletion of particles would not be possible in our case since particles represent mass elements.

With such an adaptive simulation, the granularity of matter will locally change over time. Since particles of different masses and sizes will simultaneously be present at a given time, the simulation process has to be extended to non-homogeneous particle systems. Moreover, a sudden change in the

number of particles should not produce instabilities due to local variations of mass density. Lastly, space discretization cannot be adapted without adapting time discretization in consequence, to ensure correct behavior and avoid instabilities. The remainder of this section addresses these three problems.

3.1 Simulation with a Non-Homogeneous Particle System

Individual smoothing kernels

Contrary to the model in Section 2 where the same kernel W_h was used for each particle, we now must assign an individual smoothing length to each particle. Indeed, this smoothing length h should be related to the volume of matter a particle represents: If many small particles are used in a given area to accurately discretize pressure variations, an adequately small value of h must be used. Otherwise, the pressure variations will be too much filtered. On the other hand, h must be large enough in areas sampled by few large particles. This leads to the choice of a smoothing length h that keeps the number of “neighbors” around a particle to an approximately constant value. A particle of mass m_i in a substance of rest density ρ_0 represents a volume of matter m_i/ρ_0 . Since a particle is isotropic, this volume can be seen as a sphere of radius r_i such that:

$$\frac{4}{3}\pi r_i^3 = \frac{m_i}{\rho_0} \quad (6)$$

Hence, we set the smoothing length h_i of particle i to a value proportional to r_i :

$$h = \xi \sqrt[3]{m_i/\rho_0}. \quad (7)$$

The value of the constant ξ is chosen according to the average number of neighboring particles we want to obtain around particle i .

Shooting vs. gathering

Since individual kernels are used for each particle, the term W_h used in the equation of pressure forces (equation 4) should either be replaced by the smoothing kernel W_{h_i} of the processed particle, or alternatively by the kernels W_{h_j} of its neighbors. Both ways are correct, and lead to valuable interpretations of smoothed particles [13]: in the first case, a *gathering* method is used, since a particle collects and weighs contributions from neighboring particles according to its own mass distribution; in the second case, a *shooting* method is used, i.e., a particle sums weighted contributions from other particles. In our implementation, we combine shooting and gathering. The pressure force applied on a particle i is defined as the average of the two forces derived from gathering and shooting interpretations of equation (4):

$$\mathbf{F}_i^{\nabla P} = -km_i \sum_{j \neq i} m_j \left(\frac{\rho_i - \rho_0}{\rho_i^2} + \frac{\rho_j - \rho_0}{\rho_j^2} \right) \frac{\nabla_i(W_{h_i}^{ij} + W_{h_j}^{ij})}{2} \quad (8)$$

Thus, the set of neighboring particles j with which the particle i interacts is not only controlled by the size of W_{h_i} , but is characterized by: $\|\mathbf{x}_i - \mathbf{x}_j\| < 2 \max(h_i, h_j)$. Unlike pure shooting or pure gathering, this formulation has the advantage of enforcing Newton’s third law stating that action should be opposite to reaction. All the other equations that use the smoothing kernel are modified in the same way (equations (11) and (22) for instance). Similarly, the factor h in equation (23) is replaced by $(h_i + h_j)/2$.

3.2 Differential Control of Density

The density-based deformable model reviewed in Section 2 computed density at each time step from equation (2), yielding:

$$\rho_i = \sum_j m_j W_h(\mathbf{x}_i - \mathbf{x}_j) \quad (9)$$

This expression is not appropriate in practice, especially when particles of different masses and sizes are used:

- Firstly, since the value of mass density given by equation (9) depends on the number of neighboring particles, the density of a uniform distribution of identical particles drops near boundaries. Particles would thus cluster near the surface of the material during simulations to balance this loss, giving rise to irregular equilibrium configurations.
- The perspective of an adaptive simulation yields another constraint on the way density is computed: splitting a particle into several smaller ones should not result in a sudden change of mass density in this area. If equation (9) was used, maintaining local density at a constant value during a split would be almost impossible.

We propose an alternative strategy for computing mass density during simulations. We set an initial mass density value for each particle and then use the following continuity equation for computing the variations of density over time (where \mathbf{v} stands for velocity):

$$\dot{\rho} = -\rho \nabla \cdot \mathbf{v}. \quad (10)$$

This formulation, expressing mass preservation [1], offers a differential control of the density. The divergence of velocity $\nabla \cdot \mathbf{v}$ can be approximated through [21]:

$$(\nabla \cdot \mathbf{v})_i = \frac{1}{\rho_i} \sum_{j \neq i} m_j (\mathbf{v}_j - \mathbf{v}_i) \cdot \nabla_i W_h^{ij} \quad (11)$$

Thus, as density is updated according to the local relative motions of particles inside the substance rather than from their absolute space positions, it does not generate problems near borders. Moreover, if velocities of smaller particles are set to the velocity of their parent during a split, no punctual change of local density will be produced as a result.

3.3 Correlation between Space and Time

Our aim is to find an adaptive simulation strategy that reduces computation by dynamically and locally adjusting both space and time discretizations. However, spatial and temporal resolutions are far from independent : the tinier the particle, the shorter the time step, so as to capture all deformation waves inside the material properly.

Space is crucial for both adequately compromising between accuracy and efficiency in the discrete approximation of fields and for visual purpose. Our approach is thus to adapt spatial discretization first according to the local variations of pressure and to the desired accuracy. We then adapt time accordingly in order to ensure stability and reliability. By this means, each particle has its own size and integration time step: it optimizes computation by concentrating effort when and where needed. The two next sections detail the algorithms we use for adaptively discretizing space and time.

4 Adaptive Space Discretization

Space subdivision must be locally adapted to the complexity of deformation in progress. In our model, high deformation occurs when high pressure forces are present. An intuitive idea for adaptively discretizing space and ensure stability is thus to maintain an optimized sampling of mass density: few smoothed particles will be sufficient in areas where mass density is almost constant, while more will be needed in areas where density varies much to get a better description of the density variations. A sound criterion implementing this idea needs to be defined.

4.1 Refinement

Smoothed particles rely on a Monte-Carlo approximation of integrals (equations (2) and (3)). More precisely, an importance sampling approach is used [22], since integration over mass instead of over volume is performed through the introduction of mass density. As a consequence, a simulation will be reliable *if integration errors are minimized*, i.e., if a good sampling of mass density variations is provided (see Figure 1 (b)). This leads to the following refinement criterion: A particle i is divided into smaller ones if the difference of pressure with one of its neighbors weighted by the volume it samples is too large:

$$\text{divide particle } i \text{ iff: } \exists j \text{ neighbor of particle } i / |\rho_j - \rho_i| \frac{m_i}{\rho_i} > \Delta \quad (12)$$

where Δ is a threshold value under the user's control. Since the variations of density are directly related to pressure in our model, refinement occurs where pressure forces are significant, which intuitively means that deformation is expected.

The division of a particle is performed by replacing it with a set of n smaller particles that are distributed in the volume the former one was occupying, as depicted in Figure 1(a). In practice, we limit the minimum size to a given "intrinsic granularity" for the material the user wants to simulate. In order to conserve mass and momentum, the mass m_j^{new} and the velocity \mathbf{v}_j^{new} of the smaller particles are set to:

$$m_j^{new} = \frac{m_i}{n} \quad \mathbf{v}_j^{new} = \mathbf{v}_i,$$

while h_j^{new} is computed from m_j^{new} as in equation (7).

The densities ρ_j^{new} are interpolated from the density value of the parent particle and its neighbors as depicted in 1D in Figure 1(b). With the differential control of density detailed in Section 3.2, no sudden change in the material behavior is generated: the subsequent motion just better describes the deformation in progress.

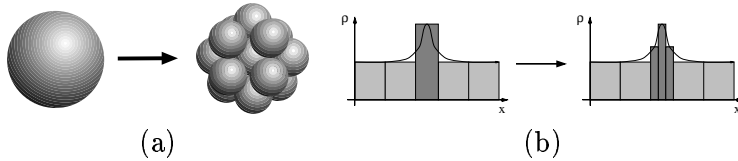


Figure 1: Refinement is performed by dividing a particle into smaller ones in areas where density varies a lot, offering an enhanced sampling.

We have experimented several values for the number of smaller particles. Using seven particles to re-sample the subdivided particle seems a reasonable trade-off between accuracy and computational time.

4.2 Simplification

Areas where density does not vary much are very stable, since deformations in the density-based model are due to differences of pressure. The criterion we use for characterizing stable areas is:

$$\text{merge particles around } i \text{ iff: } \forall j \text{ neighbor of } i, |\rho_j - \rho_i| \frac{m_i}{\rho_i} < \delta \quad (13)$$

where δ is a threshold value under the user's control.

Unlike refinement, using a criterion on density is not sufficient. As depicted in Figure 2(a), not all stable areas can be merged into a single large particle: since a particle is isotropic, the stable particles should approximately fill a sphere, otherwise the simplification process will debase the local geometric description of the substance.

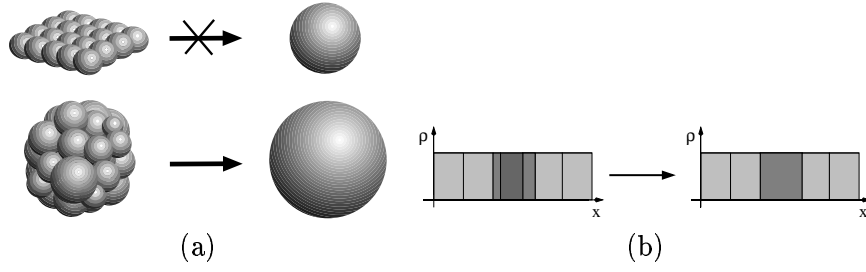


Figure 2: Simplification is performed by merging a set of neighboring particles occupying a volume close to a sphere into a large one

One way to check this property is to compute the *local inertia matrix* I_M of the set of stable particles, considering each particle as a mass point. If $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{z})$ is the center of mass of these particles, then the inertia matrix can be written:

$$I_M = \begin{pmatrix} I_{Ox} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{Oy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{Oz} \end{pmatrix}$$

with :

$$\begin{aligned} I_{Ox} &= \sum_j m_j ((y_j - \tilde{y})^2 + (z_j - \tilde{z})^2) & I_{Oy} &= \sum_j m_j ((x_j - \tilde{x})^2 + (z_j - \tilde{z})^2) \\ I_{Oz} &= \sum_j m_j ((x_j - \tilde{x})^2 + (y_j - \tilde{y})^2) & I_{xy} &= \sum_j m_j (x_j - \tilde{x})(y_j - \tilde{y}) \\ I_{yz} &= \sum_j m_j (y_j - \tilde{y})(z_j - \tilde{z}) & I_{xz} &= \sum_j m_j (x_j - \tilde{x})(z_j - \tilde{z}) \end{aligned}$$

If the three eigenvalues of this matrix are almost equal, the mass distribution is roughly spherical. Then simplifying the particles into a single one is a must. To save calculations, we simply compute the *determinant* of the inertia matrix I_M , and its *trace*. As the determinant is equal to the product of the three eigenvalues while the trace is the sum, we just check if:

$$\left(\frac{\text{trace}(I_M)}{3} \right)^3 \simeq \det(I_M) \quad (14)$$

as it ensures that all the eigenvalues, which are positive, are identical. This efficiently detects spherical arrangements without having to perform eigenvalues search.

The new particle is positioned at the center of gravity of the deleted ones, its mass is set to the sum of their masses, its velocity is computed in order to conserve the linear momentum, and its density is computed from the new mass and the volume the new particle should sample:

$$\begin{aligned} m_i^{new} &\leftarrow \sum_j m_j \\ \mathbf{x}_i^{new} &\leftarrow (\sum_j m_j \mathbf{x}_j) / m_i^{new} \\ \mathbf{v}_i^{new} &\leftarrow (\sum_j m_j \mathbf{v}_j) / m_i^{new} \\ \rho_i^{new} &\leftarrow \frac{m_i^{new}}{\sum_j (m_j / \rho_j)}. \end{aligned}$$

In our implementation, we perform a fusion test around a particle i just before integrating forces on it. Since the neighbors of the particle have just been computed, testing if particle i can merge with

its neighbors does not add much to computation: Comparisons of density are performed according to equation (13) in a loop which also computes the center of mass of the group of particles. If none of the density tests fail, we eliminate non-spherical distributions by first checking that the center of mass of the particles is close to particle i . If needed, we compute the trace and determinant of the inertia matrix and use equation (14). Refinement tests are performed within the same loop.

5 Adaptive Time Discretization

As explained in Section 4, our approach is to adapt time discretization to the current discretization of space. A solution would be to integrate the applied forces on each particles at a very small time step, according to the size of the current smallest particle. To save computation, we instead associate an *individual time step* with each particle. During an animation, a particle will integrate its equation of motion only when needed, according to its current individual time step. Between two force evaluations, its position and speed will be updated according to the latest computed acceleration, so that other smaller particles can perform their own force evaluations². Since all particles must re-synchronize every frame, all the time steps used equal the frame rate divided by a power of two, for easiness of implementation.

This section gives criteria for computing the time steps, and then summarizes the resulting simulation algorithm.

5.1 Individual Time Steps

During an animation, the time step of a particle should be sufficiently small to avoid instabilities, but not too small for efficiency. An important criterion for accurate integration is the Courant condition [22]. Principally used in finite differencing methods, this criterion states that a wave front should not *pass over* a sample point in one time step. Since our fastest pressure wave is namely the speed of sound c , given by equation (5), we can write:

$$dt_i \leq \lambda_1 \frac{h_i}{c} \quad \text{where } \lambda_1 < 1 \quad (15)$$

Moreover, the animation of the density-based model heavily relies on good approximations of speed and density sampled at particles. Since these two values may vary too quickly when a collision occurs, we add two extra conditions, that respectively bound the rates of change of speed and density during one time step:

$$dt_i \leq \lambda_2 \sqrt{\frac{h_i}{\|\mathbf{a}_i\|}} \quad dt_i \leq \frac{\lambda_3}{|(\nabla \cdot \mathbf{v})_i|} \quad (16)$$

where λ_2 and λ_3 are unitless constant, and \mathbf{a}_i is the current acceleration of particle i . Other criteria can of course be added for specific applications.

During a simulation, the value of the next time step to be used for a particle is computed each time a force evaluation has been performed, since acceleration and divergence of velocity may have changed. This is also done when a particle has been created during a refinement or simplification of space resolution. In order to synchronize the animation every dt_{frame} , the individual time step dt_i is set to the largest value $dt_{\text{frame}}/2^q$ that verifies the three criteria above.

²Action/reaction law is not exactly enforced in this case since the smaller particle reevaluates the interaction force every dt_j while the force integrated by particle i is considered as constant during a time interval of length $dt_i > dt_j$. But if time step criterion ensures that the rate dt_i is sufficient for adequately approximating motion of particle i , this problem is neither significant, nor perceived.

5.2 Animation Algorithm

An adaptive simulation as described above results in an implementation quite similar to the standard density-based model [8], except that the most expensive part of computation, i.e. the nearest neighbors search, is only performed when needed. This saves a lot of computation, as we will show in Section 7. The algorithm can be sum up as follows.

At time t :

- For each particle that currently needs force evaluations:
 1. Compute the list of neighboring particles from current positions;
 2. Test for refinement/simplification as explained in Section 4, and perform refinement/simplification if needed.
 3. Integrate forces: Use equations (8) and (22) for evaluating pressure and viscosity forces from the neighboring particle's positions and densities. Add external actions such as gravity and collision forces. Compute new acceleration from the set of current applied forces \mathbf{F}_i : $\mathbf{a}_i \leftarrow \mathbf{F}_i/m_i$.
 4. Compute the divergence of velocity of the particle from equation (11), and use equation (10) for evaluating $\dot{\rho}_i$.
 5. Use the values of acceleration and divergence of velocity for computing the next time step value dt_i of the particle, from the criteria (15) and (16).
- For all particles, update speed, position and density from respectively their current values of \mathbf{a}_i , \mathbf{v}_i and $\dot{\rho}_i$, with an Euler or Leapfrog scheme [22].
- Set t to $t + dt_{min}$, where dt_{min} is the current smallest time step needed.

We have coded the time steps with integers, according to $dt_i = dt_{frame}/2^{q_i}$, $q_i \in [0..q_{max}]$. This way, we can use boolean operations such as shifts or logical ands to efficiently check if forces evaluation is needed for a particle at time t knowing its current dt_i . We have compared performances of nearest neighbors searches in a non-homogeneous particle system using a specific octree and a fixed grid over space. Due to the large overhead of the octree, the grid has turned out to be better in all our examples of Section 7.

6 Visualization using active Implicit Surface

Producing nice synthetic images from an animation performed by particles is not trivial. When the particles are modeling gaseous phenomena, volume rendering should be used for realistic results [26]. For substances ranging from solids to liquids, rendering an iso-surface of a sum of scalar fields generated by the particles is more appropriate [20, 28]. An iso-surface easily deals with topology changes such as separations or fusions inside the substance, can be used to detect collisions with obstacles, and may be constrained to produce constant volume deformation [7]. The case of smoothed particles is even simpler, since there is no need for defining an extra scalar field: an iso-surface of mass density $S_\rho = \{\mathbf{x}/\rho(\mathbf{x}) = iso\}$ can be rendered, and this is coherent with the underlying physical model [8].

However, this solution conveys a number of disadvantages:

- Rendering an iso-mass density can be very expensive. Both direct ray-tracing or polygonization of an implicit surface [2] require binary searches of surface points along rays or along voxel edges. These searches are time consuming when the field is generated by hundreds of particles.
- Depending on the choice of the iso-value iso , unwanted volume variations can be generated during animations as stressed in [8].

- Lastly, using an iso-mass-density would produce significant and sudden visual artifacts during simulations, since the surface S_ρ would locally and suddenly change at each refinement/simplification of the particle system.

Our solution is to use an active implicit surface model, inspired from the active contour models — or “snakes” — used in Computer Vision [16]. This surface will track the iso-mass density surface S_ρ while mimicking surface tension: In this way, the artifacts due to a change of granularity will be blocked off. We review the main features of the model hereafter, and explain how it can address the problem of adaptive simulation rendering. A more detailed description of the active implicit model can be found in [6].

6.1 A discrete implicit function

Our aim is to provide a surface model convenient for handling topological changes (pieces of substance may separate or melt during the animation), but that can be more efficiently polygonized than skeleton-based implicit surfaces such as those generated by former methods [20, 7]. We thus propose to model the surface of the substance by an iso-surface of a *discrete field* (or *potential*) whose values are stored on a 3D grid. Contrary to most of the implicit surfaces used in Computer Graphics, the potential will no longer have an analytic formulation, but only sampled values from which a continuous field can be interpolated if needed. These sample values will be modified dynamically to finally animate the surface, which is defined as:

$$S(t) = \{\mathbf{x} \in \mathbb{R}^3 / f(\mathbf{x}) = 0\}$$

This discrete formulation offers several advantages. First, the evaluation cost of the potential of a point in space can be performed in constant time, *independently of the surface complexity*, via a basic interpolation of the closest values. Secondly, finding sample points on the resulting surface is quite straightforward compared to general implicit surfaces: Since we have at hand a grid of potential values, a simple spatial partitioning method (also called Marching Cubes) [30, 17] can be used. We choose a tri-linear interpolation between potential values so that the null potential between two adjacent grid nodes of opposite values can be found in a simple linear interpolation, without any need for compute-intensive root-finding methods. Moreover, such a sampling method provides a trivial polygonization of the surface from the obtained sample points. Lastly, a further advantage given by this discrete potential is *local control* of the surface. Indeed, we can act on a single potential value to affect the surface slightly and locally, without complicating the formulation.

6.2 Making the implicit surface evolve

Animating the iso-surface means modifying the discrete field values of f . To ensure continuity in the motion of the surface S , we integrate the following differential equation, obtained via a simple chain rule:

$$\frac{\partial f}{\partial t}(\mathbf{x}(t), t) = -\nabla f(\mathbf{x}(t), t) \cdot \frac{d\mathbf{x}(t)}{dt} \quad (17)$$

Put differently, if we define a *velocity* $\frac{d\mathbf{x}}{dt}$ on each point of the surface, we can numerically integrate equation (17) over time on each point of the grid, using finite differencing. This will in turn make the iso-surface evolve in this velocity field.

6.3 Choice of a velocity field

Our choice for $\frac{d\mathbf{x}(t)}{dt}$ relies on the three following properties:

1. First, the surface should track the given iso-density S_ρ of the adaptive particle system;
2. The surface should also be smooth. The current granularity of the particle system should not be visually perceived;
3. The internal volume defined by $f \geq 0$ should stay almost constant during the animation.

We thus use the equation:

$$\frac{d\mathbf{x}}{dt}(t) = \left[\alpha (\rho(\mathbf{x}) - iso) + \beta \kappa(\mathbf{x}) + \gamma \frac{(V_0 - V(t))}{V_0} \right] \mathbf{n}(\mathbf{x}) \quad (18)$$

where $\mathbf{n}(\mathbf{x}) = \nabla f(\mathbf{x}) / \|\nabla f(\mathbf{x})\|$ is the current normal to the surface at point \mathbf{x} , and α, β, γ are three parameters weighing the relative importance of the three desired properties above.

The first term in equation (18) force the surface $f = 0$ to locally inflate along its normal if it is too much inside the substance (i.e., if $\rho(\mathbf{x}) > iso$), and to locally deflate if it is too much outside. This tracking strategy is very similar to the “balloon snakes” model used in Computer Vision [5].

The second term models *surface tension*, since it minimizes curvature. Tuning it thus filters the small features of the underlying particle system, hiding the granularity.

Finally, the third term tends to keep the internal volume near a constant value V_0 , which can be useful to balance small variations of volume during deformations. The current volume is easily determined via the triangles created during the rendering stage.

6.4 Optimized implementation

The choice of a discrete formulation for the implicit field allows us different optimizations. Since only field values in the neighborhood of the surface of interest are relevant (for finite differences), the field f is only stored on few vertices of the grid surrounding the iso-surface. The other values can be thresholded at -1 for the exterior and 1 for the interior, as sketched in Figure 3. A linked list of voxels of interest, near the current surface, is dynamically maintained during the animation

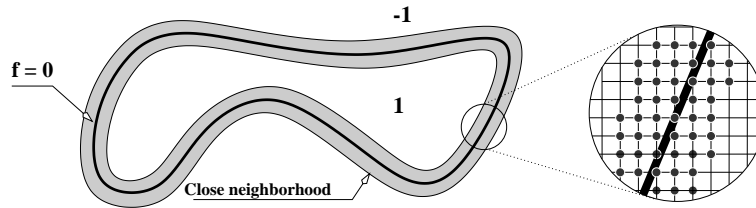


Figure 3: Neighborhood around the iso-surface for a discrete field, in 2D.

In addition to reduced storage and optimized computational cost, keeping tracks of the surface neighborhood that surrounds the iso-surface $f = 0$ greatly accelerates polygonization: The voxels that intersect the surface, already stored in a linked list, just have to be processed to approximate the local surface via triangles [2]. “Marching” from cube to cube is no longer needed.

6.5 Overview of the animation algorithm

The user first sets a size for the grid (in practice, an animation is rendered with a coarse grid first, and then with a finer one when the global visual motion is satisfactory). The field f is initialized on the grid vertex from the values of mass density of the particle system at the beginning of the animation. A list of nodes which lie in the neighborhood of the iso-surface $\rho = iso$ is created, other vertices are

thresholded, as exterior or interior.

At each time step, equation (17) is integrated on each node on the list, velocity being given by equation (18). In order to avoid the usual instabilities of this Hamilton-Jacobi differential equation [24], we compute $\rho(\mathbf{x}) - iso$ from interpolation of density values on the current surface points [6]. Every voxel in the surface neighborhood is processed, and a set of triangles approximating the iso-surface is generated [2]. The linked list of active grid nodes is dynamically updated when the surface moves from a voxel to another. In our simulations, the tubular neighborhood is about three voxel wide on each side of the surface. Time stepping is computed once again from the Courant condition [6].

With the cumulated effects of tracking and tension surface, changes of granularity are filtered and not visually perceived. Compared to conventional rendering of particles coated by implicit surfaces, our method brings several additional advantages:

- $\rho(\mathbf{x})$, which may be expensive to compute, is evaluated just once at each time step and only for each vertex of the surface neighborhood, as no binary search is performed to find the surface.
- The number of vertices in the surface neighborhood is linear with respect to the area of the surface $f = 0$. This allows us interactive rates for grids of reasonable size.
- Finally, the “reconstructed” discrete field provides a fast inside/outside test, much more efficient than any evaluation of the mass density. It means that the computation involved for the rendering will save a valuable time for collision detection for instance.

Qualitative results and computation time for generating the surface are given in the next section.

7 Results

7.1 Computing animations

Before performing a simulation, our system computes an initial discretization of the substance to animate. The desired initial shape is filled with hexagonally ordered 2-D layers of particles with the tiniest particle size according to the chosen finest granularity of simulation. All particles are assigned the same mass and initial density. Once the initial sampling has been set up, a pass of simplification is performed in order to automatically simplify the inside of the volume. The substance is then ready to be animated.

The animations are either visualized by displaying spheres of radius r_i for each particle (see equation (6)), or by using the active implicit surface of Section 6. The first visualization gives a very good perception of the refinements/simplifications that are taking place, while the second one filters them and offers fine visualization. In our implementation, collisions with obstacles are detected using the spheres. Accuracy could be improved by implementing retroaction from the implicit surface to the particles as in [7], so that the surface could be used for collision processing.

The user controls the animation by setting mainly three intuitive parameters for the substance: rest mass density, stiffness, and viscosity. The other parameters are rapidly tuned to obtain the desired accuracy/efficiency ratio. We have used the following parameters settings in standard units :

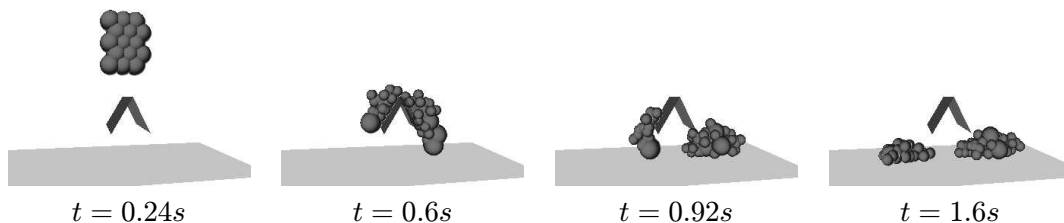
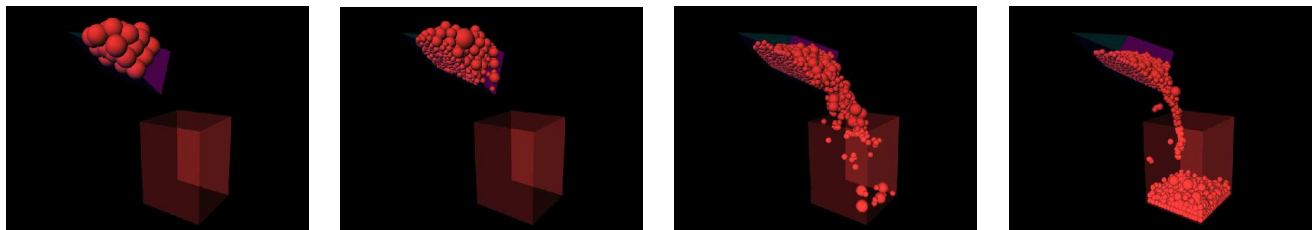


Figure 4: Substance falling over an obstacle: adaptive simulation

Figure 5: Pouring substance in a box: adaptive simulation ($t = 0s, 0.04s, 0.4s$, and $1.2s$).

Parameter	Value or Range	Meaning
ρ_0	$100 - 10^6$	Density at rest
k	$10 - 10^6$	Stiffness
η	$0.1 - 30$	Viscosity
ξ	1.35	Scope-of-influence parameter
Δ	$10^{-3} - 100$	Refinement threshold
δ	$10^{-4} - 10$	Simplification threshold
λ_1	.3	Courant number
λ_2	.5	Precise speed integration
λ_3	.005	Precise density integration
α	.5 - 50	Tracking factor
β	$10^{-4} - .1$	Surface tension factor
γ	1 - 100	Constrained volume factor

7.2 Qualitative results

Examples of adaptive simulations showing the refinement/simplification process are shown in Figures 7 and 5. As expected, refinements take place in areas of high deformations and/or contact with obstacles. Simplifications are less easily observed since they take place deeper in the material, but they play an important part in the efficiency of computations, as will be shown below.

Our qualitative comparisons between standard and adaptive simulations show that, even if the results are not exactly the same³, performing adaptive simulations does not alter visual realism. In these experiments, space resolution in standard simulations has been set to the smallest scale of the adaptives one. Such a comparison is depicted on Figure 6 for a given frame.

Visual realism is enhanced when an active implicit surface is used for coating the particles as in the example of Figure 7. The surface filters temporal changes of granularity of the particle system, as better shown in the video. Figure 8 compares a coarse and a finer versions of the active surface (obtained by changing the resolution of the grid storing the field). Grid points of the tubular neighborhood (see Section 6) are shown at the top right for the finer grid.

³Defining what “being the same” means when different numbers of particles are used would be difficult anyway.

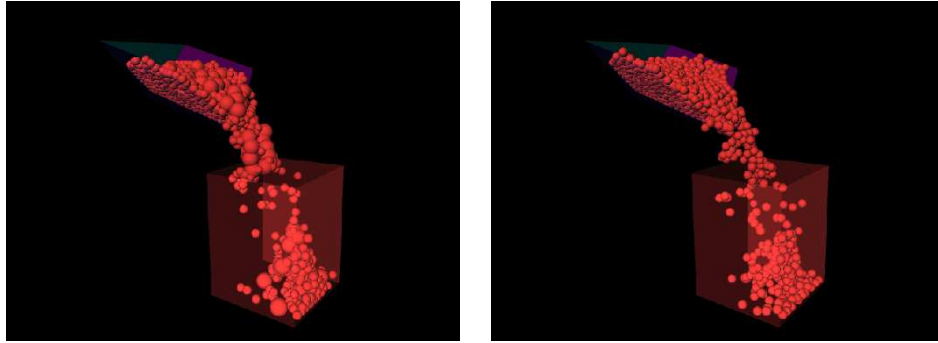


Figure 6: Comparison between a standard (left) and an adaptive (right) simulation of the animation in Figure 5 ($t = 0.56s$).

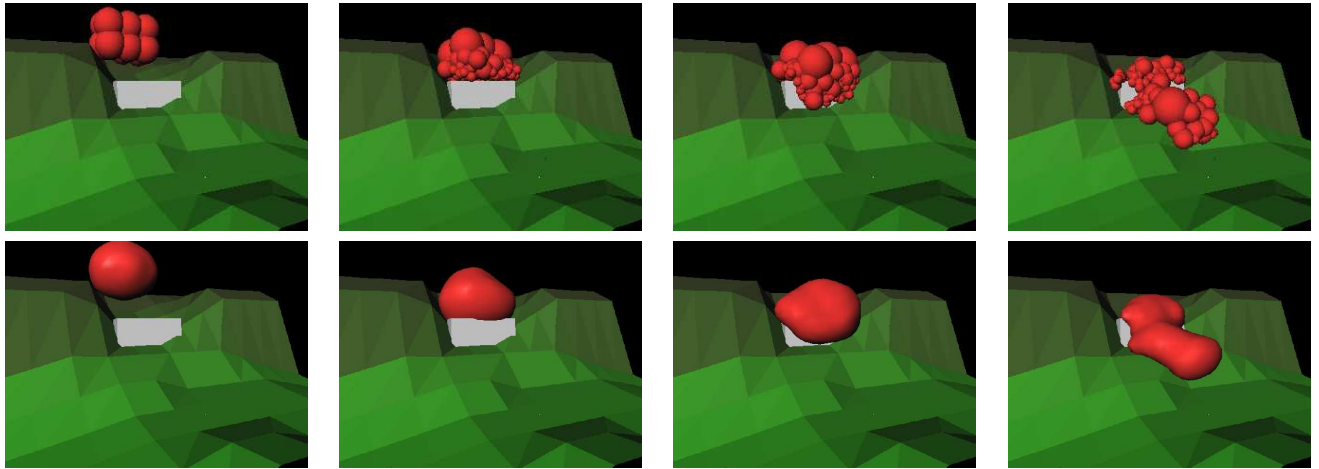


Figure 7: Substance flowing down a terrain: adaptive simulation ($t = 0.3s, 1.64s, 2.6s$ and $4.1s$).

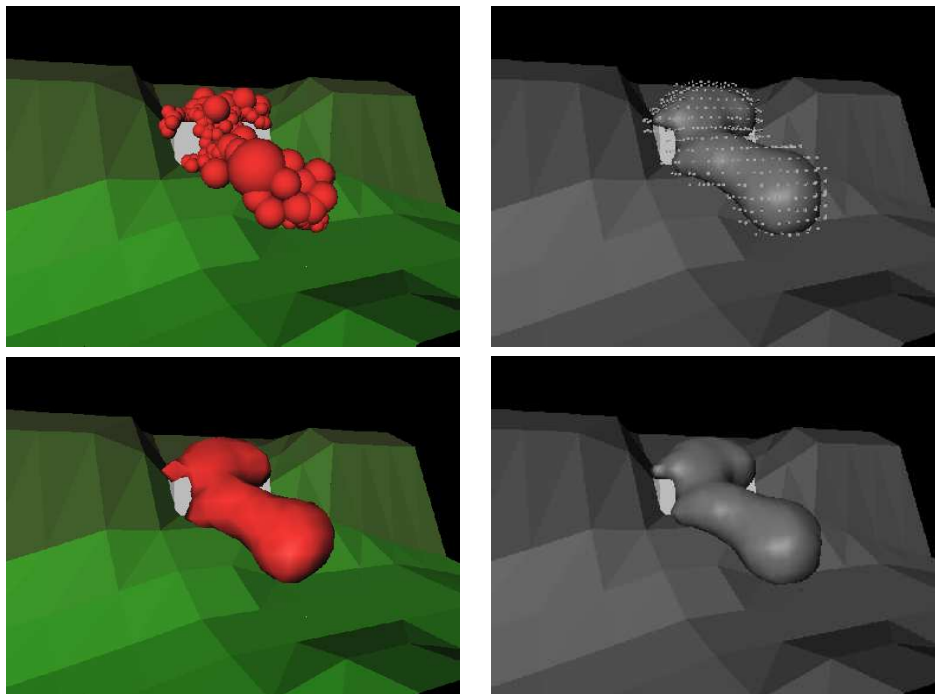


Figure 8: Comparison of two resolutions for the active implicit surface (bottom). The same frame is shown with sphere representation of particles (top left). Grid nodes of the tubular neighborhood for the surface at bottom right are depicted at top right.

7.3 Numerical results

One of the main advantages of adaptive simulation is the speed-up factor it provides. We have compared running times of standard and adaptive simulations for the animation sequences of Figure 4 and 5. Computation has been performed on a SGI O_2 R5000 workstation.

For the animation of Figure 4, the standard simulation is performed with 105 smoothed particles of mass 20 kg. The integration time step is $3.125 \cdot 10^{-4}$ (128 integrations for each frame). The total sequence has been computed in 1220 seconds, which represents an average running time of 12 seconds per frame. Using an adaptive simulation, the average time for each frame falls down to 2.9s, and the whole animation is computed in 263s. The speed-up factor over time is depicted in Figure 9.

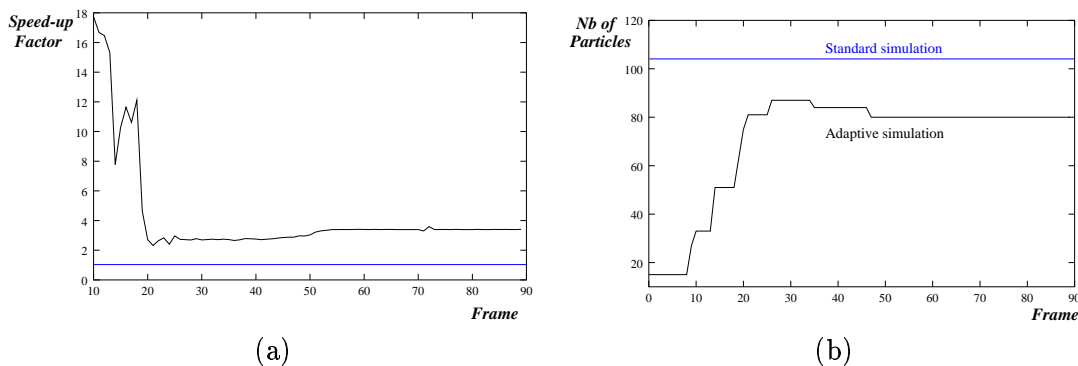


Figure 9: Numerical results for the animation of Figure 4.

Numerical results for a more complex example of Figures 5 and 6 are depicted in Figure 10. Here, the standard simulation involves 1519 smoothed particles of mass 11.2g, simulated with an integration step of 1.562510^{-4} . Each frame of the animation takes 14.9 minutes which leads to 230 hours of computation for the whole sequence. The speed-up factor we obtain when using the adaptive simulation of Figure 5 has an average value of 4. Computing the whole sequence takes 60 hours; since it is still time consuming, in practice, we first compute a coarser adaptive simulation of the same substance (see video) where the finest discretization scale is not used, yielding interactive rates (1 to 10 frames per second). Parameters tuning can be done with this coarser version (the density-based model ensures that the same material is approximated whatever the resolution). This way, the animation of Figure 5 has been computed only once with corrected parameters.

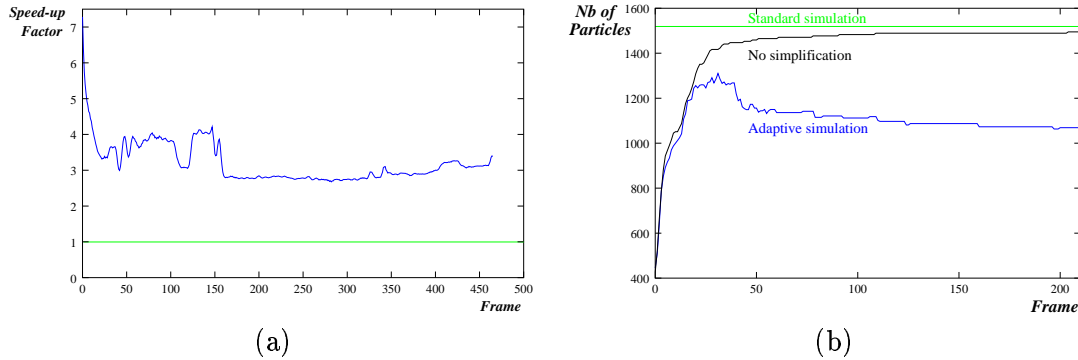


Figure 10: Numerical results for the animation of Figure 5

Generating the active implicit surface does not add much to computational time. While the adaptive animation of Figure 7 takes 1s per frame in average, computing the coarse surface of Figure 8 (left) takes 0.1s per frame, while the finer one (right) takes 0.6s per frame.

8 Conclusions & Future Work

We have presented a new algorithm for the simulation of highly deformable substances ranging from solids to liquids. It optimizes computation while avoiding divergence by adaptively refining space and time discretizations in order to preserve accuracy.

The key point is to use a deformable model specified at a macroscopic scale by few global parameters. Such a model can be approximated at any sampling resolution by Smoothed Particles, whose interaction forces are derived from the state equation that controls the model. During the simulation, the particle system refines/simplifies automatically. Our results show that it greatly improves efficiency, since the speed-up factor is at least 3 for all our test animations. The adaptive scheme does not alter visual realism and just produces another approximation of the same substance of reference. Since there is no longer a need to specify space and time resolutions beforehand, it also reduces the animator's work. In practice, a good way to tune the animation is to compute it first with a coarse adaptive simulation, which runs at interactive rates. When the animator is satisfied with the global behavior, she/he increases the ratio between accuracy and efficiency, and runs a finer adaptive simulation. Such design of motion by successive refinements cannot be done with conventional physically-based simulation systems.

Animations with particles cannot be used without a convenient model for visualization. The problem is exacerbated with adaptive simulations, since the surface model must filter the temporal changes of granularity of the particle system. Our solution relies on an active implicit surface inspired from "snakes" used in Computer Vision. The surface provides a smooth coating of the particle system, that

deforms and may separate into pieces over time. Moreover, our method provides a control over volume variations and an efficient polygonalization.

Current work includes the implementation of retroaction between the surface and the particles as in [7]. The surface would then detect collisions and transmit response forces to the closest particles, providing a fully coupled hybrid model. We are also experimenting with materials defined by other state equations, including equations that incorporate temperature. This could be used for modeling phase-changing materials such as lava.

References

- [1] G.K Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1973.
- [2] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.
- [3] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In *Graphics Interface'97*, pages 1–8, Kelowa, British Columbia, 1997.
- [4] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, August 1996.
- [5] Laurent D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing : Image Understanding*, 53(2):211–218, March 1991.
- [6] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for computer animation. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, 1998.
- [7] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 287–290. ACM SIGGRAPH, Addison Wesley, August 1995. Los Angeles, CA.
- [8] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: A new approach for animating highly deformable bodies. In *7th Eurographics Workshop on Animation and Simulation*, Poitiers, France, September 1996.
- [9] Gary A. Dilts. Equivalence of the SPH Method and a Space-Time Galerkin Moving Particle Method. Technical Report LA-UR 96-134, Los Alamos National Laboratory, January 1996.
- [10] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.
- [11] Nick Foster and Dimitri Metaxas. Modeling the motion of a hot, turbulent gas. *Computer Graphics*, pages 181–188, 1997. Proceedings of SIGGRAPH'97 (Los Angeles, California).
- [12] Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. Proceedings SIGGRAPH '91 in Las Vegas (USA).
- [13] Lars Hernquist and Neal Katz. TREESPH: A unification of SPH with the hierarchical tree method. *App. J. Supp.*, 70:419, 1989.
- [14] Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.
- [15] Dave Hutchinson, Martin Preston, and Terry Hewitt. Adaptive refinement for mass/spring simulation. In *7th Eurographics Workshop on Animation and Simulation*, pages 31–45, Poitiers, France, September 1996.
- [16] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes : Active contour models. In *1st Conference on Computer Vision*, pages 321,331, London, U.K., June 1988.
- [17] William Lorensen and Harvey Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

- [18] Annie Luciani, Arash Habibi, Alexis Vapillon, and Yves Duroc. A physical model of turbulent fluids. In *6th Eurographics Workshop on Animation and Simulation*, Maastricht, Netherlands, September 1995.
- [19] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. A unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.
- [20] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 89. This paper also appeared in SIGGRAPH'89 Course notes number 30.
- [21] J. J. Monaghan. Smoothed Particle Hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543, 1992.
- [22] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C, second edition*. Cambridge University Press, New York, USA, 1992.
- [23] Trina M. Roy. Physically-based fluid modeling using smoothed particle hydrodynamics. *Master's thesis, University of Illinois*, 1988.
- [24] James A. Sethian. *Level Set Methods*. Cambridge Press, 1996.
- [25] François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995. (a preliminary version appeared in the fifth Eurographics workshop on rendering, Darmstadt, Germany, June 1994).
- [26] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 129–136. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.
- [27] Demetri Terzopoulos, John Platt, and Kurt Fleisher. Heating and melting deformable models (from goop to glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, June 1989.
- [28] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface'91*, pages 255–262, Calgary, AL, June 1991.
- [29] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94.
- [30] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.

A Mathematical bases

Suppose we have a field f defined over all space. We can calculate a mean value within a spatial interval (more precisely, a convolution) with:

$$\langle f(\mathbf{r}) \rangle = \int_{space} f(\mathbf{r}') W_h(\mathbf{r} - \mathbf{r}') d\mathbf{r}' \quad (19)$$

where $W_h(\mathbf{r})$ is a smoothing kernel, and h the smoothing length specifying the extent of the averaging volume. $W_h(\mathbf{r})$ is peaked about $\mathbf{r} = 0$ so that it tends to the Dirac delta function as $h \rightarrow 0$, keeping its integral normalized to 1.

Suppose now that our kernel W_h has a finite support. Then we can approximate any derivative of f easily. For instance, we have by definition in 1D:

$$\langle \frac{\partial f}{\partial x}(r) \rangle = \int_{-\infty}^{+\infty} \frac{\partial f}{\partial x}(r') W_h(r - r') dr'.$$

Integrating by parts, this yields:

$$\int_{-\infty}^{+\infty} \frac{\partial f}{\partial x}(r') W_h(r - r') dr' = [f(r') W_h(r - r')]_{r=-\infty}^{r=+\infty} - \int_{-\infty}^{+\infty} f(r') \cdot (-1) \cdot \frac{\partial W_h}{\partial x}(r - r') dr',$$

so that we can write:

$$\langle \frac{\partial f}{\partial x}(r) \rangle = \int_{-\infty}^{+\infty} f(r') \frac{\partial W_h}{\partial x}(r - r') dr'. \quad (20)$$

It proves that there is no need to know analytical derivatives to calculate their mean values.

B Smoothing Kernel

We use the same kernel as in [8], depicted in Figure 11 :

$$W_h(\mathbf{x}) = \frac{15}{\pi(4h)^3} \begin{cases} (2 - \frac{\|\mathbf{x}\|}{h})^3 & \text{if } 0 \leq \|\mathbf{x}\| \leq 2h \\ 0 & \text{if } \|\mathbf{x}\| > 2h \end{cases} \quad (21)$$

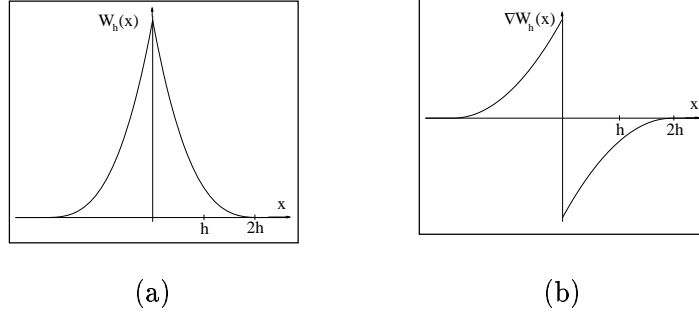


Figure 11: The smoothing kernel and its gradient in 1D.

As the Gaussian and spline kernels, this kernel guarantees stability since its Fourier transform rapidly falls with wave numbers [21]. Its advantage, as shown in [8], is to produce pairwise interaction forces between particles that are qualitatively the same as the Lennard-Jones attraction/repulsion forces used in previous work [20, 27, 28, 19, 7].

C Derivation of Pressure Forces

A natural way to write the pressure force exerted on particle i in SPH formalism is:

$$\mathbf{F}_i^{\nabla P} = -\nabla P_i dV = -m_i \frac{\nabla P_i}{\rho_i} = \frac{m_i}{\rho_i} \sum_j m_j \frac{P_j}{\rho_j} \nabla W_h(\mathbf{x} - \mathbf{x}_j).$$

However, the action-reaction principle would not be enforced with this formulation ($m_i \nabla P_i / \rho_i$ is not equal to $m_j \nabla P_j / \rho_j$ for $i \neq j$). To symmetrize the pressure forces, we use (as in [8]) the derivation rule $\nabla P / \rho = \nabla (P / \rho) + P \nabla \rho / \rho^2$ to write:

$$\frac{\nabla P_i}{\rho_i} = \sum_{j \neq i} m_j \frac{P_j}{\rho_j^2} \nabla_i W_h^{ij} + \frac{P_i}{\rho_i^2} \sum_{j \neq i} m_j \nabla_i W_h^{ij}.$$

The pressure force applied on a particle is then:

$$\mathbf{F}_i^{\nabla P} = -m_i \sum_{j \neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i W_h^{ij}.$$

D Viscosity

As in the SPH literature [21, 13], we model viscosity by adding an extra force representing internal friction to the sum of external actions exerted on a particle. This force is computed from the distance and relative speed of neighboring particles. The general equation we use is:

$$\mathbf{F}_i^{viscosity} = -\eta \sum_{j \neq i} m_i m_j \Pi_{ij} \nabla_i W_h^{ij} \quad (22)$$

where the factor η controls the amount of viscosity, and where Π_{ij} is given by:

$$\Pi_{ij} = \begin{cases} \frac{hc(\mathbf{v}_{ij} \cdot \mathbf{x}_{ij})}{\bar{\rho}_{ij}(\mathbf{x}_{ij}^2 + h^2/100)} & \text{if } \mathbf{v}_{ij} \cdot \mathbf{x}_{ij} < 0 \\ 0 & \text{if } \mathbf{v}_{ij} \cdot \mathbf{x}_{ij} \geq 0 \end{cases} \quad (23)$$

$$\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j \quad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j \quad \bar{\rho}_{ij} = (\rho_i + \rho_j)/2$$



Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399