A material point method for snow simulation

Alexey Stomakhin[⋆†] Craig Schroeder[†] Lawrence Chai[⋆] Joseph Teran[⋆†] Andrew Selle[⋆]

[†]University of California Los Angeles [⋆]Walt Disney Animation Studios

Abstract

Snow is a challenging natural phenomenon to visually simulate. While the graphics community has previously considered accumulation and rendering of snow, animation of snow dynamics has not been fully addressed. Additionally, existing techniques for solids and fluids have difficulty producing convincing snow results. Specifically, *wet* or *dense* snow that has both solid- and fluid-like properties is difficult to handle. Consequently, this paper presents a novel snow simulation method utilizing a user-controllable elasto-plastic constitutive model integrated with a hybrid Eulerian/Lagrangian Material Point Method. The method is continuum based and its hybrid nature allows us to use a regular Cartesian grid to automate treatment of self-collision and fracture. It also naturally allows us to derive a grid-based semi-implicit integration scheme that has conditioning independent of the number of Lagrangian particles. We demonstrate the power of our method with a variety of snow phenomena including complex character interactions.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: material point, snow simulation, physically-based modeling

Links: DL PDF WEB

1 Introduction

Snow dynamics are amazingly beautiful yet varied. Whether it is *powder snow* fluttering in a skier's wake, foot steps shattering an *icy snow* crust or even *packing snow* rolled into balls to make a snowman, it is snow's rich repertoire that makes it simultaneously compelling for storytelling and infuriatingly difficult to model on a computer. Artists typically use simpler techniques combined in various ways to achieve snow effects [Kim and Flores 2008; Coony et al. 2010; Klohn et al. 2012], but these approaches are often tweaked for one type of snow. This suggests the need for a specialized solver that handles difficult snow behaviors in a single solver.

Specialized solvers for specific phenomena are frequently used in graphics and computational physics because achieving maximum resolution (and thus visual quality) requires efficiency. While a fluid simulator can produce solid-like elastic effects (and vice versa), it is not the most optimal strategy. When solids and fluids are needed simultaneously, researchers have developed two-way coupled systems to get good accuracy and performance for both phenomena. Unfortunately, snow has continuously varying phase effects, sometimes behaving as a rigid/deforming solid and sometimes behaving as a fluid. Thus, instead of discrete coupling we must simultaneously handle a continuum of material properties efficiently in the same domain, even though such a solver may not be most efficient for a single discrete phenomenon.

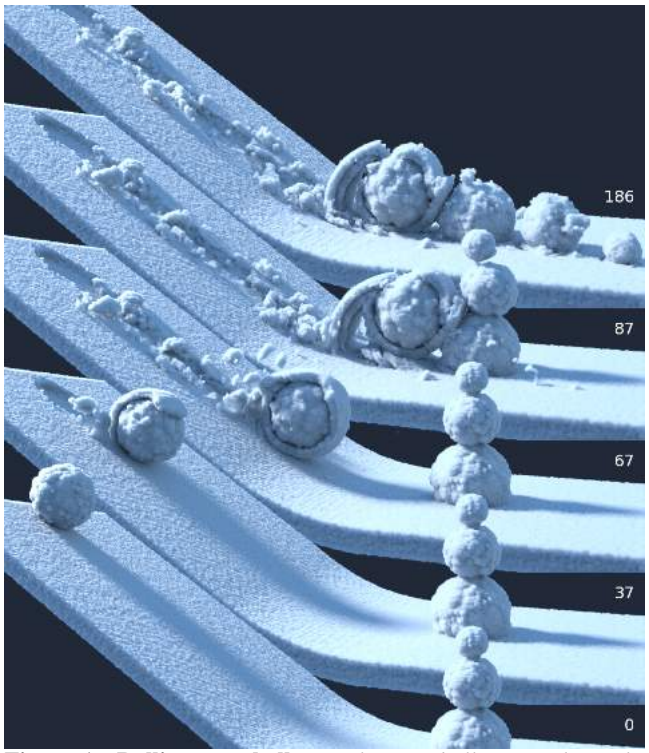

Figure 1: Rolling snowball. *As the snowball moves down the hill, compressed snow sticks, demonstrating that we can handle so-called packing snow effect.* ©*Disney.*

We present two main contributions that achieve these aims. First, we develop a semi-implicit Material Point Method (MPM) [Sulsky et al. 1995] specifically designed to efficiently treat the wide range of material stiffnesses, collisions and topological changes arising in complex snow scenes. To our knowledge, this is the first time MPM has been used in graphics. MPM methods combine Lagrangian material particles (points) with Eulerian Cartesian grids. Notably, there is no inherent need for Lagrangian mesh connectivity. Many researchers in graphics have experimented with hybrid grid and particle methods. For example, [Zhu and Yang 2010] simulate sand as a fluid using a PIC/FLIP incompressible fluid technique. In fact, MPMs were designed as a generalization of the PIC/FLIP solvers to computational solids. As with PIC/FLIP, MPMs implicitly handle self-collision and fracture with the use of the background Eulerian grid. This is essential given the many topological changes exhibited by practical snow dynamics. Our second contribution is a novel snow constitutive model designed for intuitive user control of practical snow behavior. This is also designed to achieve our goal of

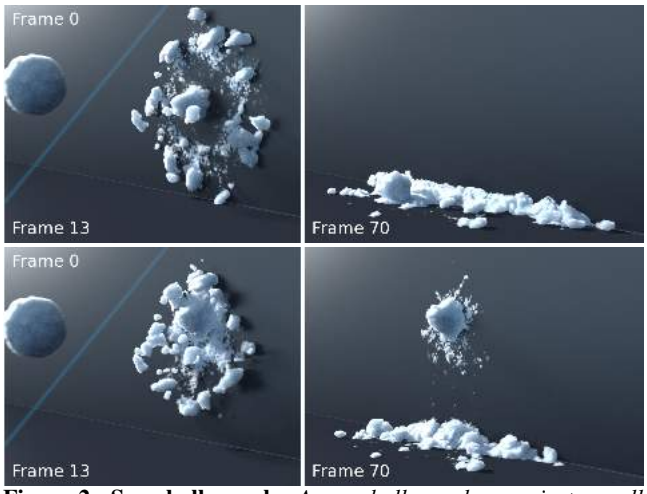

Figure 2: Snowball smash. *A snowball smashes against a wall with sticky (bottom) and non-sticky (top) collisions.* ©*Disney.*

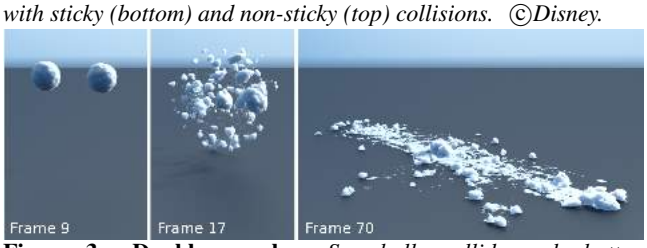

Figure 3: Double smash. *Snowballs collide and shatter.* ©*Disney.*

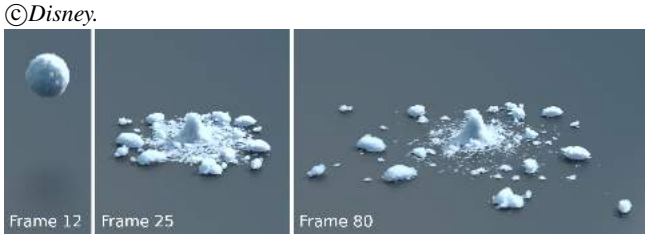

Figure 4: Snowball drop. *A basic snowball hitting the ground.* ©*Disney.*

describing the many phases of snow behavior with one constitutive relation. To do this, we borrow from the vast engineering literature on snow and demonstrate that an elasto-plastic treatment is an effective means of handling the transition between many different behaviors including flowing, clumping, breaking and more.

The paper continues with a discussion of related work in Section 2. In Section 3 we present the basics of MPM, and in Sections 4 through 9 we present details of the method. Finally, we discuss results and conclusions in Sections 10 and 11.

2 Related work

Geometric snow modeling The bulk of graphics snow research is devoted to modeling accumulation [Feldman and O'Brien 2002; Fearing 2000a; Fearing 2000b; Nishita et al. 1997]. These techniques can efficiently and accurately create snow-covering effects but neglect treatment of snow dynamics. Some authors have extended these techniques to handle rapid animation and interactions with external objects [Pla-Castells et al. 2006; Zhu and Yang 2010; Chanclou et al. 1996; Marechal et al. 2010; Hinks and Museth 2009]. Often these methods use simplified modeling primitives like height-fields (e.g. [Sumner et al. 1999]), and these are now a popular techniques for games and feature films. Additionally, [Kim and Lin 2003; Kim et al. 2006] simulate the related phenomena of ice and frost formation.

Granular materials Traditionally, snow in graphics is thought of as a granular material, but we are not aware of any graphics paper that specifically simulates snow dynamics. Granular material behavior is mostly determined by inter-granular friction, and there are two main ways of handling this: considering each grain and its interactions explicitly or considering grains in an aggregate continuum fashion. The first graphics papers approximating granular materials mostly did the former by using simplified particle or rigid body systems [Miller and Pearce 1989; Luciani et al. 1995; Milenkovic 1996], and recently researchers have continued to improve such techniques [Bell et al. 2005; Alduán et al. 2009]. Even so, the challenge is to retain efficiency as the number of grains increases, leading other researchers to apply simplified continuum models [Zhu and Bridson 2005a; Lenaerts and Dutré 2009; Narain et al. 2010; Alduán and Otaduy 2011; Ihmsen et al. 2012] to good effect. In particular, [Zhu and Bridson 2005a] introduced a FLIP-based method for simulating sand as an incompressible fluid. [McAdams et al. 2009] used incompressible FLIP to model hair collision behavior, although modeling friction and cohesion accurately was problematic. Subsequently, [Narain et al. 2010] modified the incompressibility constraint to avoid cohesion errors. Similarly, the MPM method was designed to extend FLIP to solid mechanics problems that require compressibility. It is interesting to note that when considering granular material as a fluid, friction is approximated with viscosity.

Elasto-plastic continuum modeling The computer graphics work on elasto-plastic simulation is relevant because we treat snow using a constitutive model. [Terzopoulos and Fleischer 1988] pioneered modeling of deformable plasticity and fracture in graphics. [O'Brien et al. 2002] considered ductile fracture using remeshing and a multiplicative plasticity model. [Pauly et al. 2005] use a point-based elasto-plastic model to handle fracture. [Irving et al. 2004; Stomakhin et al. 2012] consider improving robustness of large (and plastic) deformation by correcting constitutive models for tetrahedral inversion, and this approach is augmented with remeshing in [Bargteil et al. 2007]. [Chao et al. 2010] further improve deformable models and in particular show the importance of handling non-linear derivatives correctly. [Goktekin et al. 2004; Keiser et al. 2005; Wojtan and Turk 2008] simulate non-Newtonian fluids with grids, points, and triangle mesh based solvers, respectively. Recently [Levin et al. 2011] rediscretized elasticity on an Eulerian regular grid, which is similar to the grid used by the material point method. Unlike many of the above techniques, we avoid using a persistent topology, and instead use points as our dominant representation.

Engineering modeling of snow There is extensive engineering literature related to the modeling and simulation of snow [Gray and Male 1981]. Although the complex mechanical behavior of snow strongly depends on numerous physical conditions, we found that an elasto-plastic constitutive relation worked well for generating

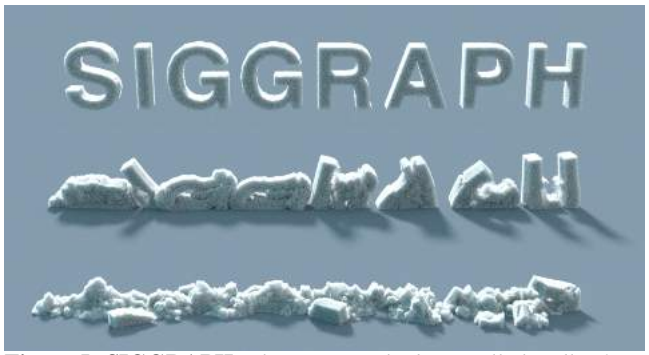

Figure 5: SIGGRAPH. *The MPM method naturally handles fracture, giving us interesting naturalistic shapes.* ©*Disney.*

Figure 6: Castle destruction. *Modeled structures like this castle can be destroyed using our method.* ©*Disney.*

realistic dynamics for a wide range of visual phenomena. This representation, as well as finite-element-based discretization, is very common in the engineering literature [Meschke et al. 1996; Cresseri and Jommi 2005; St Lawrence and Bradley 1975; Dutykh et al. 2011; Cresseri et al. 2010; Brown 1980; Nicot 2004].

3 Paper Overview

As discussed in the introduction, snow dynamics modeling is difficult due to snow's variability, usually stemming from environmental factors (freshness, water/ice content, etc.). Our model ignores root causes, and we instead concentrate on deriving an empirical model based on phenomenological observations. Even so, our snow constitutive model is based on theory and models devised for engineering applications. The model is kept efficient, allowing us to capture sufficient geometric detail with tractable computation time.

The material point method is the center of our technique. At its core, MPM relies on the *continuum approximation*, avoiding the need to model every snow grain. While an MPM method typically uses a Cartesian grid to make topology changes and self-collisions automatic, it outperforms purely Eulerian methods by tracking mass (and other conserved quantities) through non-dissipative Lagrangian particles (like SPH). Unlike SPH, however, MPM uses the grid as an efficient continuum *scratch-pad* which avoids high valence communication patterns derived from nearest-neighbor queries. See Figure 7 for an illustration of the interplay between the grid and particles and Section 4.1 for details.

Our motivation for choosing MPM is that it is better able to handle the dynamics of snow. This is analogous to how a rigid body simulation is the most efficient way to handle infinite stiffness and incompressibility compared to using a large stiffness directly on a FEM mesh solver. The constitutive properties central to snow are *volume preservation*, *stiffness*, *plasticity*, *fracture*, and we summarize various methods' abilities to handle them in Table 1.

Volume preservation in snow is important even though, unlike a liquid, it is compressible. Instead, snow has varying resistance to volume change, which we model in our method similarly to a normal mesh-based solid simulation. Stiffness is also important, and while MPM cannot do this as well as mesh-based elasticity (the deformation gradient is less accurate), it is more effective than grid-based elasticity as the deformation gradient is not dissipative and remains synchronized with positions. Plasticity and fracture are also handled well by MPM, and this is what makes the method desirable for snow simulation. MPM is almost ideal for plasticity because the inaccuracies of the deformation gradient accumulate as artificial plasticity. While grid-based fluids work well for plasticity [Goktekin et al. 2004], MPM is better at conserving angular momentum. While mesh-based FEM methods can handle plasticity [Bargteil et al. 2007], remeshing is required with extreme deformation. By contrast, MPM need only track the unmeshed particles. We note MPM's gains in plasticity and fracture come at the cost of reduced elastic accuracy, a good tradeoff for snow.

Method	Volume Preservation	Stiffness	Plasticity	Fracture
Reeve particles	-	-	-	-
Rigid bodies	⋆⋆	⋆⋆	-	⋆
Mesh-based solids	⋆	⋆⋆⋆	⋆⋆	⋆
Grid-based fluids	⋆⋆⋆	⋆	⋆⋆	⋆⋆⋆
SPH	⋆	⋆	⋆	⋆⋆⋆
MPM	⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆

Table 1: *Comparison between various methods of simulation on four properties that are important for snow.*

4 Material point method

A body's deformation can be described as a mapping from its undeformed configuration $\boldsymbol{X}$ to its deformed configuration $\boldsymbol{x}$ by $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X})$, which yields the deformation gradient $\boldsymbol{F} = \partial\boldsymbol{\phi}/\partial\boldsymbol{X}$. Deformation $\boldsymbol{\phi}(\boldsymbol{X})$ changes according to conservation of mass, conservation of momentum and the elasto-plastic constitutive relation

$$\frac{D\rho}{Dt} = 0, \quad \rho\frac{D\boldsymbol{v}}{Dt} = \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g}, \quad \boldsymbol{\sigma} = \frac{1}{J}\frac{\partial\Psi}{\partial\boldsymbol{F}_E}\boldsymbol{F}_E^T,$$

where ρ is density, t is time, $\boldsymbol{v}$ is velocity, $\boldsymbol{\sigma}$ is the Cauchy stress, $\boldsymbol{g}$ is the gravity, Ψ is the elasto-plastic potential energy density, $\boldsymbol{F}_E$ is the elastic part of the deformation gradient $\boldsymbol{F}$ and $J = \det(\boldsymbol{F})$. We will discuss details of the constitutive model in Section 5.

The basic idea behind the material point method is to use particles (material points) to track mass, momentum and deformation gradient. Specifically, particle p holds position $\boldsymbol{x}_p$, velocity $\boldsymbol{v}_p$, mass m_p, and deformation gradient $\boldsymbol{F}_p$. The Lagrangian treatment of these quantities simplifies the discretization of the $\frac{D\rho}{Dt}$ and $\rho\frac{D\boldsymbol{v}}{Dt}$ terms. However, the lack of mesh connectivity between particles complicates the computation of derivatives needed for stress-based force evaluation. This is remedied with the use of a regular background Eulerian grid. Interpolating functions over this grid are used to discretize the $\nabla\cdot\boldsymbol{\sigma}$ terms in the standard FEM manner using the weak form. We use dyadic products of one-dimensional cubic B-splines as our grid basis functions as in [Steffen et al. 2008]

$$N_{\boldsymbol{i}}^h(\boldsymbol{x}_p) = N(\tfrac{1}{h}(x_p - ih))N(\tfrac{1}{h}(y_p - jh))N(\tfrac{1}{h}(z_p - kh)),$$

where $\boldsymbol{i} = (i,j,k)$ is the grid index, $\boldsymbol{x}_p = (x_p, y_p, z_p)$ is the

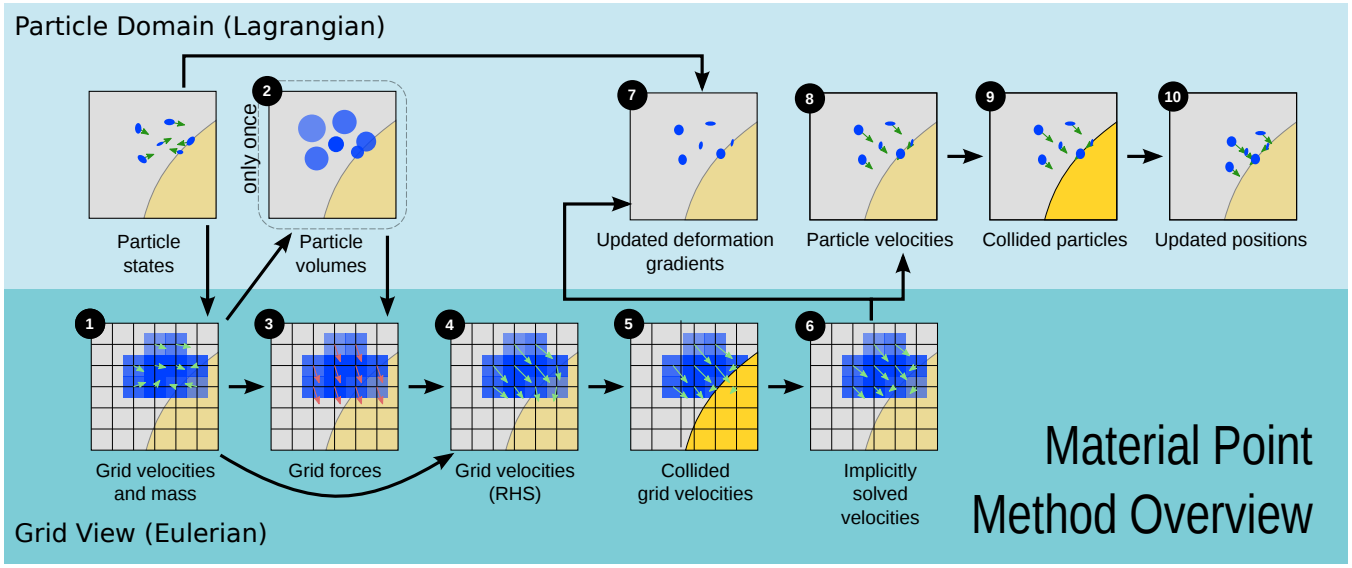

Figure 7: *An overview of the material point method (MPM). The top and the bottom rows are steps that operate on particles while the middle depicts grid-based operations.*

evaluation position, h is the grid spacing and

$$N(x) = \begin{cases} \frac{1}{2}|x|^3 - x^2 + \frac{2}{3}, & 0 \le |x| < 1 \\ -\frac{1}{6}|x|^3 + x^2 - 2|x| + \frac{4}{3}, & 1 \le |x| < 2 \\ 0, & \text{otherwise} \end{cases}.$$

For more compact notation, we will use $w_{ip} = N_i^h(\boldsymbol{x}_p)$ and $\nabla w_{ip} = \nabla N_i^h(\boldsymbol{x}_p)$. These interpolation functions naturally compute forces at the nodes of the Eulerian grid. Therefore, we must first transfer the mass and momentum from the particles to the grid so that we can update the velocities at the grid nodes. This updated velocity is then transferred back to the particles in either a FLIP or PIC type manner. The transfer process is done using the interpolating weights w_{ip}.

4.1 Full method

Here we outline the full update procedure (visually shown in Figure 7).

1. **Rasterize particle data to the grid.** The first step is to transfer mass from particles to the grid. The mass is transferred using the weighting functions $m_i^n = \sum_p m_p w_{ip}^n$. Velocity also should be transferred to the grid, but weighting with w_{ip}^n does not result in momentum conservation. Instead, we use normalized weights for velocity $\boldsymbol{v}_i^n = \sum_p \boldsymbol{v}_p^n m_p w_{ip}^n / m_i^n$. This contrasts with most incompressible FLIP implementations.

2. **Compute particle volumes and densities.** *First timestep only.* Our force discretization requires a notion of a particle's volume in the initial configuration. We can estimate a cell's density as m_i^0/h^3, which we can weight back to the particle as $\rho_p^0 = \sum_i m_i^0 w_{ip}^0 / h^3$. We can now estimate a particle's volume as $V_p^0 = m_p/\rho_p^0$.

3. **Compute grid forces** using equation (6) with $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i$.

4. **Update velocities on grid** to $\boldsymbol{v}_i^\star$ using equation (10).

5. **Grid-based body collisions** on $\boldsymbol{v}_i^\star$ as described in Section 8.

6. **Solve the linear system** in equation (9) for semi-implicit integration. For explicit time integration, simply let $\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^\star$.

7. **Update deformation gradient.** The deformation gradient for each particle is updated as $\boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1})\boldsymbol{F}_p^n$, where we have computed $\nabla \boldsymbol{v}_p^{n+1} = \sum_i \boldsymbol{v}_i^{n+1}(\nabla w_{ip}^n)^T$. Section 7 gives a detailed description of the update rule for elastic and plastic parts of $\boldsymbol{F}$.

8. **Update particle velocities.** Our new particle velocities are $\boldsymbol{v}_p^{n+1} = (1-\alpha)\boldsymbol{v}_{\text{PIC}p}^{n+1} + \alpha \boldsymbol{v}_{\text{FLIP}p}^{n+1}$, where the PIC part is $\boldsymbol{v}_{\text{PIC}p}^{n+1} = \sum_i \boldsymbol{v}_i^{n+1} w_{ip}^n$ and the FLIP part is $\boldsymbol{v}_{\text{FLIP}p}^{n+1} = \boldsymbol{v}_p^n + \sum_i (\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n) w_{ip}^n$. We typically used $\alpha = 0.95$.

9. **Particle-based body collisions** on $\boldsymbol{v}_p^{n+1}$ as detailed in Section 8.

10. **Update particle positions** using $\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \boldsymbol{v}_p^{n+1}$.

5 Constitutive model

Snow material behavior is complicated by the fact that it contains a combination of water and ice which strongly affects its behavior. One might be tempted to think that snow behaves like sand. However, one major difference is that snow is typically compressible while sand is not. In addition, its behavior changes dramatically with a number of environmental factors including temperature, humidity, density and snow age, making snow constitutive modeling a challenging and open research problem. We refer the reader to [Nicot 2004] for a thorough discussion of the many approaches in the engineering literature and [Gray and Male 1981] as a general snow reference.

We found the methodology employed in [Meschke et al. 1996] to be most relevant to computer graphics because it is concerned with the large strains typical of visually compelling scenes. In this work, the authors use a specially designed finite-strain multiplicative plasticity law employing the Drucker-Prager plasticity model [Drucker and Prager 1952]. They couple this with a hyperelastic dependence of the Kirchoff stress on the elastic part of the multiplicative decomposition of the deformation gradient. While this model is designed to match the stress-strain relation of snow under a number of realistic conditions, we found that a more simplified treatment of finite-strain multiplicative plasticity coupled with a common graphics model for hyperelasticity was sufficient for vi-

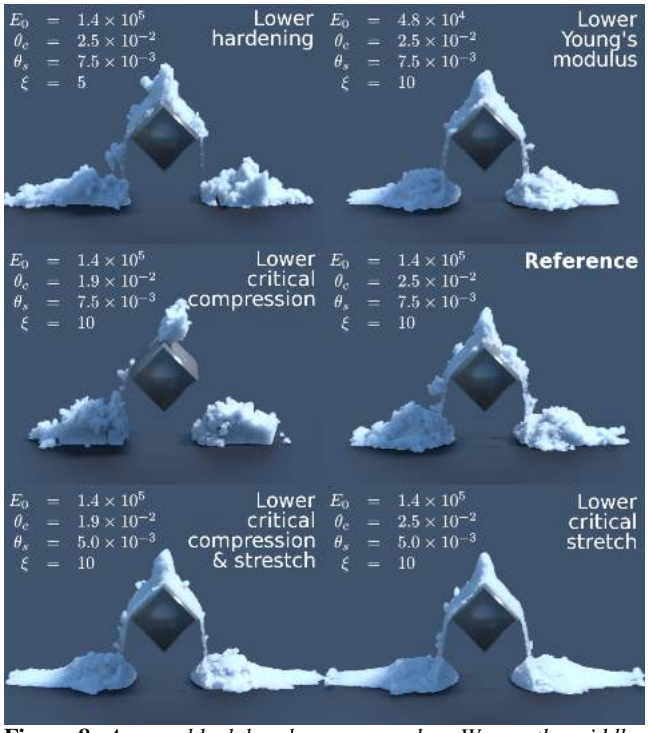

Figure 8: *A snow block breaks over a wedge. We use the middle-right image as a reference and show how changing different parameters in our model affects the look and dynamics of the material.* ©*Disney.*

sual realism. The most salient features of our approach are the use of principal stretches rather than principal stresses in defining our plastic yield criteria as well as a simplification of hardening behavior that only requires modification of the Lamé parameters in the hyperelastic energy density. While principal-stress-based plasticity is more appropriate for physical accuracy, principal-stretch-based yield gives the user more control over the visual behavior of the simulation.

In multiplicative plasticity theory it is customary to separate $\boldsymbol{F}$ into an elastic part $\boldsymbol{F}_E$ and a plastic part $\boldsymbol{F}_P$ so that $\boldsymbol{F} = \boldsymbol{F}_E\boldsymbol{F}_P$. We define our constitutive model in terms of the elasto-plastic energy density function

$$\Psi(\boldsymbol{F}_E, \boldsymbol{F}_P) = \mu(\boldsymbol{F}_P)\|\boldsymbol{F}_E - \boldsymbol{R}_E\|_F^2 + \frac{\lambda(\boldsymbol{F}_P)}{2}(J_E - 1)^2, \quad (1)$$

with the elastic part given by the fixed corotated energy density from [Stomakhin et al. 2012] and the Lamé parameters being functions of the plastic deformation gradients

$$\mu(\boldsymbol{F}_P) = \mu_0 e^{\xi(1-J_P)} \text{ and } \lambda(\boldsymbol{F}_P) = \lambda_0 e^{\xi(1-J_P)}, \quad (2)$$

where $J_E = \det \boldsymbol{F}_E$, $J_P = \det \boldsymbol{F}_P$, $\boldsymbol{F}_E = \boldsymbol{R}_E\boldsymbol{S}_E$ by the polar decomposition, λ_0, μ_0 are the initial Lamé coefficients and ξ is a dimensionless plastic hardening parameter. Additionally we define the portion of deformation that is elastic and plastic using the singular values of the deformation gradient. We define a critical compression θ_c and stretch θ_s as the thresholds to start plastic deformation (or fracture). Namely, the singular values of $\boldsymbol{F}_E$ are restricted to the interval $[1 - \theta_c, 1 + \theta_s]$.

Our material is elastic in the regime of small deformations as dictated by the $\boldsymbol{F}_E$ dependence in (1). When the deformation exceeds a critical threshold (either stretch or compress) it starts deforming plastically as described in more detail in Section 7. This also affects

Parameter	Notation	Value
Critical compression	θ_c	2.5×10^{-2}
Critical stretch	θ_s	7.5×10^{-3}
Hardening coefficient	ξ	10
Initial density (kg/m^3)	ρ_0	4.0×10^2
Initial Young's modulus (Pa)	E_0	1.4×10^5
Poisson's ratio	ν	0.2

Table 2: *In our model we found these parameters to be a useful starting point for producing simulations.*

the material properties in accordance with (2), making it stronger under compression (packing) and weaker under stretch (fracture), allowing us to achieve realistic snow phenomena.

To simulate different types of snow, we found the following intuition useful. θ_c and θ_s determine *when* the material starts breaking (larger = chunky, smaller = powdery). The hardening coefficient determines *how fast* the material breaks once it is plastic (larger = brittle, smaller = ductile). *Dry* and *powdery* snow has smaller critical compression and stretch constants, while the opposite is true for *wet* and *chunky* snow. *Icy* snow has a higher hardening coefficient and Young's modulus, with the opposite producing *muddy* snow. See Figure 8 for examples of snow variation and Table 2 for a list of generic parameters.

6 Stress-based forces and linearization

The total elastic potential energy can be expressed in terms of the energy density Ψ as

$$\int_{\Omega^0} \Psi(\boldsymbol{F}_E(\boldsymbol{X}), \boldsymbol{F}_P(\boldsymbol{X}))d\boldsymbol{X}, \quad (3)$$

where Ω^0 is the undeformed configuration of the material. The MPM spatial discretization of the stress-based forces is equivalent to differentiation of a discrete approximation of this energy with respect to the Eulerian grid node material positions. However, we do not actually deform the Eulerian grid so we can think of the change in the grid node locations as being determined by the grid node velocities. That is, if $\boldsymbol{x_i}$ is the position of grid node $\boldsymbol{i}$, then $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{x_i} + \Delta t \boldsymbol{v_i}$ would be the deformed location of that grid node given the current velocity $\boldsymbol{v_i}$ of the node. If we refer to the vector of all grid nodes $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ as $\hat{\boldsymbol{x}}$, then the MPM approximation to the total elastic potential can be written as

$$\Phi(\hat{\boldsymbol{x}}) = \sum_p V_p^0 \Psi(\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}}), \boldsymbol{F}_{Pp}^n),$$

where V_p^0 is the volume of material originally occupied by particle p, $\boldsymbol{F}_{Pp}^n$ is the plastic part of $\boldsymbol{F}$ at particle p at time t^n and $\hat{\boldsymbol{F}}_{Ep}$ is the elastic part which is related to $\hat{\boldsymbol{x}}$ as in [Sulsky et al. 1995] as

$$\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}}) = \left(\boldsymbol{I} + \sum_{\boldsymbol{i}} (\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x_i})(\nabla w_{\boldsymbol{i}p}^n)^T\right)\boldsymbol{F}_{Ep}^n. \quad (4)$$

With this convention, the MPM spatial discretization of the stress-based forces is given as

$$-\boldsymbol{f_i}(\hat{\boldsymbol{x}}) = \frac{\partial \Phi}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}}}(\hat{\boldsymbol{x}}) = \sum_p V_p^0 \frac{\partial \Psi}{\partial \boldsymbol{F}_E}(\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}}), \boldsymbol{F}_{Pp}^n)(\boldsymbol{F}_{Ep}^n)^T \nabla w_{\boldsymbol{i}p}^n. \quad (5)$$

That is, $\boldsymbol{f_i}(\hat{\boldsymbol{x}})$ is the force on grid node $\boldsymbol{i}$ resulting from elastic stresses. This is often written in terms of the Cauchy stress $\boldsymbol{\sigma}_p = \frac{1}{J_p^n}\frac{\partial \Psi}{\partial \boldsymbol{F}_E}(\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}}), \boldsymbol{F}_{Pp}^n)(\boldsymbol{F}_{Ep}^n)^T$ as

$$\boldsymbol{f_i}(\hat{\boldsymbol{x}}) = -\sum_p V_p^n \boldsymbol{\sigma}_p \nabla w_{\boldsymbol{i}p}^n, \quad (6)$$

Figure 10: Snowplow. *The characteristic cylindrical tube spray is created by the snowplow.* ©*Disney.*

where $V_p^n = J_p^n V_p^0$ is the volume of the material occupied by particle p at time t^n.

We highlight this relation of the MPM spatial discretization to the elastic potential because we would like to evolve our grid velocities $\boldsymbol{v_i}$ implicitly in time. With this convention, we can take an implicit step on the elastic part of the update by utilizing the Hessian of the potential with respect to $\hat{\boldsymbol{x}}$. The action of this Hessian on an arbitrary increment $\delta\boldsymbol{u}$ can be expressed as

$$-\delta \boldsymbol{f_i} = \sum_{\boldsymbol{j}} \frac{\partial^2 \Phi}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}}(\hat{\boldsymbol{x}}) \delta \boldsymbol{u_j} = \sum_p V_p^0 \boldsymbol{A}_p (\boldsymbol{F}_{Ep}^n)^T \nabla w_{\boldsymbol{i}p}^n, \quad (7)$$

where

$$\boldsymbol{A}_p = \frac{\partial^2 \Psi}{\partial \boldsymbol{F}_E \partial \boldsymbol{F}_E}(\boldsymbol{F}_E(\hat{\boldsymbol{x}}), \boldsymbol{F}_{P_p}^n) : \left(\sum_{\boldsymbol{j}} \delta \boldsymbol{u_j} (\nabla w_{\boldsymbol{j}p}^n)^T \boldsymbol{F}_{Ep}^n \right) \quad (8)$$

and the notation $\boldsymbol{A} = \boldsymbol{\mathcal{C}} : \boldsymbol{D}$ is taken to mean $A_{ij} = \mathcal{C}_{ijkl} D_{kl}$ with summation implied on indices kl. See the accompanying technical report for details of the differentiation.

6.1 Semi-implicit update

We think of the elasto-plastic response as defined from the material positions of the Eulerian grid nodes $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{x_i} + \Delta t \boldsymbol{v_i}$. However, as noted in the previous section, we never deform this grid. Therefore, we can think of $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(\boldsymbol{v})$ as defined by $\boldsymbol{v}$. With this in mind, we use the following notation $\boldsymbol{f_i^n} = \boldsymbol{f_i}(\hat{\boldsymbol{x}}(\boldsymbol{0}))$, $\boldsymbol{f_i^{n+1}} = \boldsymbol{f_i}(\hat{\boldsymbol{x}}(\boldsymbol{v}^{n+1}))$ and $\frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} = -\frac{\partial \boldsymbol{f_i^n}}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} = -\frac{\partial \boldsymbol{f_i}}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}}(\hat{\boldsymbol{x}}(\boldsymbol{0}))$.

Using these derivatives, we form our implicit update using $\boldsymbol{v_i^{n+1}} = \boldsymbol{v_i^n} + \Delta t m_{\boldsymbol{i}}^{-1}((1-\beta)\boldsymbol{f_i^n} + \beta \boldsymbol{f_i^{n+1}}) \approx \boldsymbol{v_i^n} + \Delta t m_{\boldsymbol{i}}^{-1}(\boldsymbol{f_i^n} + \beta \Delta t \sum_{\boldsymbol{j}} \frac{\partial \boldsymbol{f_i^n}}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \boldsymbol{v_j^{n+1}})$. This leads to a (mass) symmetric system to solve for $\boldsymbol{v_i^{n+1}}$

$$\sum_{\boldsymbol{j}} \left(\boldsymbol{I} \delta_{\boldsymbol{ij}} + \beta \Delta t^2 m_{\boldsymbol{i}}^{-1} \frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \right) \boldsymbol{v_j^{n+1}} = \boldsymbol{v_i^\star}, \quad (9)$$

where the right hand side is

$$\boldsymbol{v_i^\star} = \boldsymbol{v_i^n} + \Delta t m_{\boldsymbol{i}}^{-1} \boldsymbol{f_i^n} \quad (10)$$

and β chooses between explicit ($\beta = 0$), trapezoidal ($\beta = \frac{1}{2}$), and backward Euler ($\beta = 1$).

7 Deformation gradient update

We start of by temporarily defining $\hat{\boldsymbol{F}}_{Ep}^{n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1}) \boldsymbol{F}_{Ep}^n$ as in (4) and $\hat{\boldsymbol{F}}_{Pp}^{n+1} = \boldsymbol{F}_{Pp}^n$, so that initially all the changes get attributed to the elastic part of the deformation gradient

$$\boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1}) \boldsymbol{F}_{Ep}^n \boldsymbol{F}_{Pp}^n = \hat{\boldsymbol{F}}_{Ep}^{n+1} \hat{\boldsymbol{F}}_{Pp}^{n+1}. \quad (11)$$

The next step is to take the part of $\hat{\boldsymbol{F}}_{Ep}^{n+1}$ that exceeds the critical deformation threshold and push it into $\hat{\boldsymbol{F}}_{Pp}^{n+1}$. We compute the singular value decomposition $\hat{\boldsymbol{F}}_{Ep}^{n+1} = \boldsymbol{U}_p \hat{\boldsymbol{\Sigma}}_p \boldsymbol{V}_p^T$ and then clamp the singular values to the permitted range $\boldsymbol{\Sigma}_p = \text{clamp}\big(\hat{\boldsymbol{\Sigma}}_p, [1-\theta_c, 1+\theta_s]\big)$. The final elastic and plastic components of the deformation gradient are computed as

$$\boldsymbol{F}_{Ep}^{n+1} = \boldsymbol{U}_p \boldsymbol{\Sigma}_p \boldsymbol{V}_p^T \quad \text{and} \quad \boldsymbol{F}_{Pp}^{n+1} = \boldsymbol{V}_p \boldsymbol{\Sigma}_p^{-1} \boldsymbol{U}_p^T \boldsymbol{F}_p^{n+1}. \quad (12)$$

It can be easily verified that $\boldsymbol{F}_p^{n+1} = \boldsymbol{F}_{Ep}^{n+1} \boldsymbol{F}_{Pp}^{n+1}$.

8 Body Collisions

We process collisions against collision bodies twice each time step. The first time is on the grid velocity $\boldsymbol{v_i^\star}$ immediately after forces are applied to grid velocities. In the case of semi-implicit integration, this contributes to the right hand side of the linear system, and degrees of freedom corresponding to the colliding grid nodes are projected out during the solve. We apply collisions once more to particle velocities $\boldsymbol{v}_p^{n+1}$ just before updating positions to account for the minor discrepancies between particle and grid velocities due to interpolation. In each case, collision processing is performed the same way. All of our collisions are inelastic.

Collision objects are represented as level sets, which makes collision detection ($\phi \leq 0$) trivial. In case of a collision the local normal $\boldsymbol{n} = \nabla \phi$ and object velocity $\boldsymbol{v}_{co}$ are computed. First, the particle/grid velocity $\boldsymbol{v}$ is transformed into the reference frame of the collision object, $\boldsymbol{v}_{rel} = \boldsymbol{v} - \boldsymbol{v}_{co}$. If the bodies are separating ($v_n = \boldsymbol{v}_{rel} \cdot \boldsymbol{n} \geq 0$), then no collision is applied. Let $\boldsymbol{v}_t = \boldsymbol{v}_{rel} - \boldsymbol{n} v_n$ be the tangential portion of the relative velocity. If a sticking impulse is required ($\|\boldsymbol{v}_t\| \leq -\mu v_n$), then we simply let $\boldsymbol{v}'_{rel} = \boldsymbol{0}$, where the prime indicates that the collision has been applied. Otherwise, we apply dynamic friction, and $\boldsymbol{v}'_{rel} = \boldsymbol{v}_t + \mu v_n \boldsymbol{v}_t / \|\boldsymbol{v}_t\|$, where μ is the coefficient of friction. Finally, we transform the collided relative velocity back into world coordinates with $\boldsymbol{v}' = \boldsymbol{v}'_{rel} + \boldsymbol{v}_{co}$.

We used two types of collision objects: rigid and deforming. In the rigid case, we store a stationary level set and a potentially time-varying rigid transform, which we can use to compute ϕ, $\boldsymbol{n}$, and $\boldsymbol{v}_{co}$ at any point. In the deforming case, we load level set key frames and interpolate them similarly to [Selle et al. 2008] using $\phi(\boldsymbol{x}, t + \gamma \Delta t) = (1-\gamma)\phi(\boldsymbol{x} - \gamma \Delta t \boldsymbol{v}_{co}, t) + \gamma \phi(\boldsymbol{x} + (1-\gamma)\Delta t \boldsymbol{v}_{co}, t + \Delta t)$, except we compute the velocity as $\boldsymbol{v}_{co} = (1-\gamma)\boldsymbol{v}(\boldsymbol{x}, t) + \gamma \boldsymbol{v}(\boldsymbol{x}, t + \Delta t)$ instead of the average velocity.

Finally, we utilize a sort of *sticky* collision in situations where we want snow to stick to vertical or under-hanging surfaces. In this case, Coulomb friction is insufficient since the normal relative velocity would be zero (vertical) or positive (under-hanging and separating due to gravity). We achieve this effect by setting $\boldsymbol{v}'_{rel} = \boldsymbol{0}$ unconditionally for collisions against these surfaces.

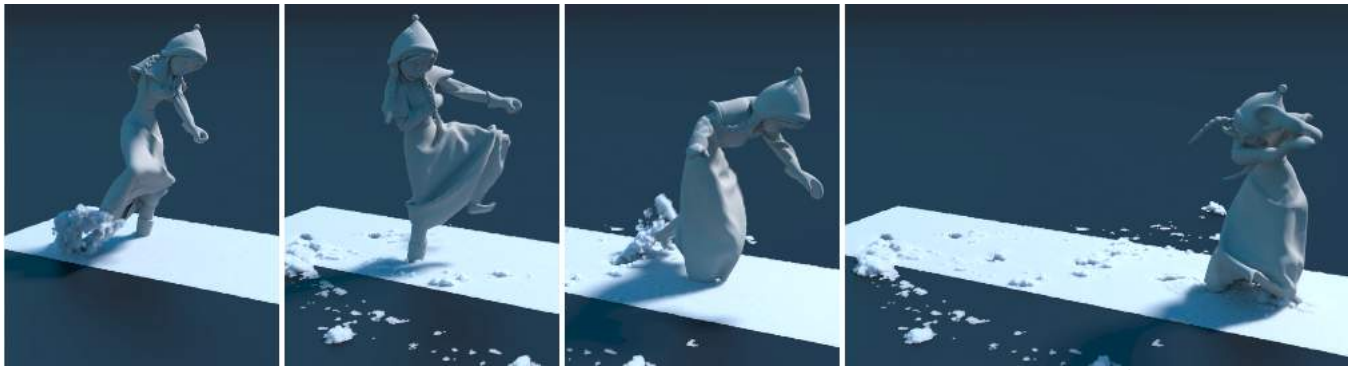

Figure 9: Walking character. *A character stepping through snow produces an interesting structure.* ©*Disney.*

9 Rendering

Researchers have measured scattering properties by applying scattering and radiative transport theory (see [Wiscombe and Warren 1980]), and this has been popular in graphics as well (see [Nishita et al. 1997]). Our discrete Cartesian grid measures density relative to the material points, giving us a way of showing visual variation between loose and tightly packed snow. This gives us a rendering advantage over a surface or a purely point-based method.

At render time, we rasterize the final simulated material points to the simulation grid using the same kernels from Section 4, but improved anti-aliasing could be obtained with better kernels or a completely different grid. . We employ a volumetric path tracer to solve the volume scattering equation using a Henyey-Greenstein phase function that approximates the Mie scattering theory of ice crystals. We typically use a mean-cosine of $g = 0.5$ to obtain forward scattering, an extinction coefficient $\sigma_t = 724\text{m}^{-1}$ and scattering albedo $\sigma_s/\sigma_t = [0.9, 0.95, 1.0]$ where σ_s is the scattering coefficient.

10 Results

We have simulated a variety of examples that demonstrate the power of our method. Our constitutive model's combination of compressibility, plasticity and hardening automatically handles fracture and *packing snow's* characteristic *sticky effect*. In particular, we show a variety of snowball simulations illustrating fracture in Figures 2, 3 and 4. We can also handle sculpted initial snow shapes fracturing as seen in Figures 5, 6 and 12. Snow's *sticky effect* is produced when originally disparate pieces of snow are compressed together which we demonstrate with a growing snowball clobbering a snowman in Figure 1.

Next we demonstrate the characteristic tumbling motion of snow being plowed in Figure 10. The simulation was performed on a translating grid to save computation. Also, we show the practicality of our method for computer animation by simulating snow interacting with animated characters. In particular, Figure 9 shows a character trudging through snow, and Figure 11 shows a character digging in snow with an ice ax.

Table 3 lists the simulation times and resolutions for each of the examples. For all of our examples we randomly seeded particles into the volumes we needed to fill in with snow and gave them the same initial parameters. We found that using $4 - 10$ particles per grid cell (for initially packed snow) produced plausible results. In addition, we found that we did not need to perform any reseeding of particles, and we also optimized grid operations to occur only on nodes where particles' interpolation radii overlapped. Thus our computation remained proportional to the number of particles and equivalently the number of occupied grid cells.

In some cases we used spatially varying constitutive parameters defined per particle to get more realistic results. For instance, the snowballs were made harder and heavier on the outside with stiffness varied with a noise pattern to get chunky fracture. This mimics our experiments with snowballs in the real world. Similarly, for the walking character, the top layer was stiffer and crunchier to mimic the effect of a night-time temperature drop.

Though simple, the explicit update scheme requires very restrictive time steps for stability; it would often require $\Delta t \simeq 10^{-5}$ to get plausible simulation behavior. By contrast, our semi-implicit method is less restrictive, allowing $\Delta t \simeq 0.5 \times 10^{-3}$ for all of the examples presented in this paper. The semi-implicit update step yields a (mass) symmetric system (9) which we solved using the conjugate residual method. In practice we found only 10-30 iterations were necessary with no preconditioning independent of grid resolution or number of particles.

11 Discussion and Conclusion

Comparison to FLIP. Incompressible FLIP [Zhu and Bridson 2005b] has become an extremely popular method in graphics for simulating liquids for many of the same reasons MPM is useful: it is simple to implement, it conserves mass, and it hybridizes grids and particles so that the best features of each representation are gained. The basic essence of both approaches is the use of a background grid as a *scratch-pad* for calculations that involve non-local transfer of information to accelerate and stabilize computation. The main difference between the two methods is that incompressible FLIP enforces incompressibility as a constraint whereas MPM allows dynamics governed by more general constitutive models. [Zhu and Bridson 2005b] integrates some notion of stress and strain (together with a rigid grouping technique) to get some constitutive modeling of sand; however, it is not as versatile as the MPM method. One can view MPM [Sulsky et al. 1995] and Incompressible FLIP [Zhu and Bridson 2005b] as generalizations of FLIP [Brackbill and Ruppel 1986] beyond its original use in compressible flow. Interestingly, Zhu and Bridson mention MPM in their paper but discount it as being too slow. Our method addresses this criticism of MPM by using semi-implicit time integration and parallelism.

Limitations. Our model has generated a large number of compelling examples, but there remains much work to be done. A lot of our parameters were tuned by hand, and it would be interesting to calibrate to measured models. We also neglect interactions with the air, which are important for powder snow and avalanches. Aliasing at render time is sometimes a difficulty, and this could be handled by better filtering or employing more sophisticated rasterization techniques. For example, we hope to experiment with the anisotropic kernel approach of [Yu and Turk 2010]. We are also interested in extending our work to sand simulation or even gen-

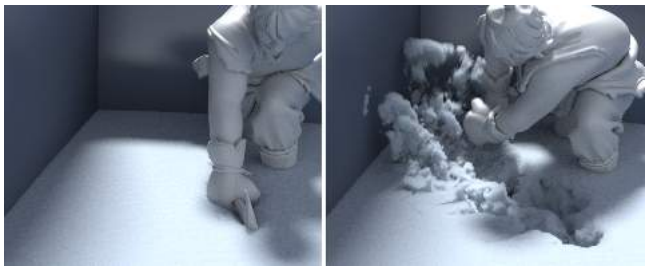

Figure 11: Character digging. *The vicious motion of a pick creates an interesting irregular corrugation in the snow.* ©*Disney.*

eral fracture. Moreover, using adaptive sparse grids would save memory for uninfluenced nodes which would save memory but not affect computation as we already do not iterate over such nodes. Similarly, employing simulation level of detail techniques could reduce run times in areas where the snow has settled. We are also interested in employing hardware acceleration via CPU SIMD or GPGPU techniques.

Conclusion. We have presented a constitutive model and simulation technique for snow simulation. Our method demonstrates a wide variety of snow behaviors, and in particular, it is the first method to handle the richness of dense and wet snow. Additionally, the material point method presents an interesting new technique for continuum mechanics that will likely inspire additional research in graphics.

Acknowledgments

Hammad Mazhar internship focusing on particle constraint snow provided inspiration for this work. We also thank the anonymous reviewers and Aleka McAdams for suggestions. UCLA authors were partially supported by NSF (DMS-0502315, DMS-0652427, CCF-0830554, DOE (09-LR-04-116741-BERA), ONR (N000140310071, N000141010730, N000141210834), Intel STC-Visual Computing Grant (20112360) as well as a gift from Disney Research. We also appreciate the support at the studio from Dayna Meltzer, Rajesh Sharma, Dan Candela, and Andy Hendrickson.

References

ALDUÁN, I., AND OTADUY, M. 2011. SPH granular flow with friction and cohesion. In *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 25–32.

ALDUÁN, I., TENA, A., AND OTADUY, M. 2009. Simulation of high-resolution granular media. In *Proc. of Congreso Español de Informática Gráfica*, vol. 1.

Example	Particles	Grid	min/frame
Snowball drop	3.0×10^5	$600 \times 300 \times 600$	5.2
Snowball smash	3.0×10^5	$200 \times 240 \times 600$	7.3
Double smash	6.0×10^5	$800 \times 300 \times 800$	13.3
Snowplow	3.9×10^6	$150 \times 50 \times 300$	2.1
Rolling snowball	7.2×10^6	$200 \times 240 \times 470$	35.7
SIGGRAPH	7.5×10^5	$780 \times 120 \times 220$	4.7
The end	5.8×10^5	$700 \times 120 \times 210$	3.8
Castle destruction	1.6×10^6	$360 \times 160 \times 560$	6.0
Walking character	3.0×10^6	$370 \times 120 \times 300$	15.7
Character digging	3.1×10^6	$280 \times 110 \times 340$	25.8

Table 3: *Example particle counts, resolutions and simulation times. Simulations were performed on an 8-core Intel Xeon X5550 2.67GHz machine.*

BARGTEIL, A., WOJTAN, C., HODGINS, J., AND TURK, G. 2007. A finite element method for animating large viscoplastic flow. In *ACM Trans. on Graph.*, vol. 26, 16.

BELL, N., YU, Y., AND MUCHA, P. 2005. Particle-based simulation of granular materials. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics symposium on Comp. animation*, 77–86.

BRACKBILL, J., AND RUPPEL, H. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. of Comp. Physics 65*, 2, 314–343.

BROWN, R. 1980. A volumetric constitutive law for snow based on a neck growth model. *J. of Appl. Phys. 51*, 1, 161–165.

CHANCLOU, B., LUCIANI, A., AND HABIBI, A. 1996. Physical models of loose soils dynamically marked by a moving object. In *Comp. Anim.'96. Proc.*, 27–35.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. *ACM Trans. on Graph. 29*, 4, 38.

COONY, I., HUTCHINS, D., LEE, K., HARRIS, M., MOLINDER, T., AND SUONG, K. 2010. Prep and landing Christmas in July: The effects snow process. In *ACM SIGGRAPH 2010 talks*.

CRESSERI, S., AND JOMMI, C. 2005. Snow as an elastic viscoplastic bonded continuum: a modelling approach. *Italian Geotechnical J. 4*, 43–58.

CRESSERI, S., GENNA, F., AND JOMMI, C. 2010. Numerical integration of an elastic–viscoplastic constitutive model for dry metamorphosed snow. *Intl. J. for Num. and Anal. Meth. in geomechanics 34*, 12, 1271–1296.

DRUCKER, D., AND PRAGER, W. 1952. Soil mechanics and plastic analysis or limit design. *Quaterly of App. Math. 10*, 157–165.

DUTYKH, D., ACARY-ROBERT, C., AND BRESCH, D. 2011. Mathematical modeling of powder-snow avalanche flows. *Studies in Appl. Math. 127*, 1, 38–66.

FEARING, P. 2000. Computer modelling of fallen snow. In *Proc. of the 27th annual conf. on Comp. Graph. and interactive techniques*, 37–46.

FEARING, P. 2000. *The computer modelling of fallen snow*. PhD thesis, University of British Columbia.

FELDMAN, B., AND O'BRIEN, J. 2002. Modeling the accumulation of wind-driven snow. In *ACM SIGGRAPH 2002 conf. abstracts and applications*, 218–218.

GOKTEKIN, T., BARGTEIL, A., AND O'BRIEN, J. 2004. A method for animating viscoelastic fluids. In *ACM Trans. on Graph.*, vol. 23, 463–468.

GRAY, D., AND MALE, D. 1981. *Handbook of snow: principles, processes, management & use*. Pergamon Press.

HINKS, T., AND MUSETH, K. 2009. Wind-driven snow buildup using a level set approach. In *Eurographics Ireland Workshop Series*, vol. 9, 19–26.

IHMSEN, M., WAHL, A., AND TESCHNER, M. 2012. High-resolution simulation of granular material with SPH. In *Workshop on Virtual Reality Interaction and Phys. Sim.*, 53–60.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Comp. animation*, 131–140.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRÉ, P., AND GROSS, M. 2005. A unified lagrangian approach to solid-fluid animation. In *Point-Based Graph., 2005. Eurographics/IEEE VGTC Symp. Proc.*, 125–148.

KIM, T.-Y., AND FLORES, L. 2008. Snow avalanche effects for Mummy 3. In *ACM SIGGRAPH 2008 talks*.

KIM, T., AND LIN, M. 2003. Visual simulation of ice crystal growth. In *Proc 2003 ACM SIGGRAPH/Eurographics Symp. Comput. Anim.*, 86–97.

KIM, T., ADALSTEINSSON, D., AND LIN, M. 2006. Modeling ice dynamics as a thin-film Stefan problem. In *Proc. 2006 ACM SIGGRAPH/Eurographics Symp. Comput. Anim.*, 167–176.

KLOHN, K., O'BRIEN, M., SPELTZ, T., AND WICHITSRIPORNKUL, T. 2012. Building the snow footprint pipeline on Brave. In *ACM SIGGRAPH 2012 talks*.

LENAERTS, T., AND DUTRÉ, P. 2009. Mixing fluids and granular materials. In *Comp. Graph. Forum*, vol. 28, 213–218.

LEVIN, D., LITVEN, J., JONES, G., SUEDA, S., AND PAI, D. 2011. Eulerian solid simulation with contact. *ACM Trans. on Graph. 30*, 4, 36.

LUCIANI, A., HABIBI, A., AND MANZOTTI, E. 1995. A multiscale physical model of granular materials. In *Graphics Interface*, 136–136.

MARECHAL, N., GUERIN, E., GALIN, E., MERILLOU, S., AND MERILLOU, N. 2010. Heat transfer simulation for modeling realistic winter sceneries. *Comp. Graph. Forum 29*, 2, 449–458.

MCADAMS, A., SELLE, A., WARD, K., SIFAKIS, E., AND TERAN, J. 2009. Detail preserving continuum simulation of straight hair. *ACM Trans. on Graphics 28*, 3, 62.

MESCHKE, G., LIU, C., AND MANG, H. 1996. Large strain finite-element analysis of snow. *J. of Engng. Mech. 122*, 7, 591–602.

MILENKOVIC, V. 1996. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. In *Proc. of the 23rd annual conf. on Comp. Graph. and interactive techniques*, 129–136.

MILLER, G., AND PEARCE, A. 1989. Globular dynamics: A connected particle system for animating viscous fluids. *Comp. & Graph. 13*, 3, 305–309.

NARAIN, R., GOLAS, A., AND LIN, M. 2010. Free-flowing granular materials with two-way solid coupling. In *ACM Trans. on Graph.*, vol. 29, 173.

NICOT, F. 2004. Constitutive modelling of snow as a cohesive-granular material. *Granular Matter 6*, 1, 47–60.

NISHITA, T., IWASAKI, H., DOBASHI, Y., AND NAKAMAE, E. 1997. A modeling and rendering method for snow by using metaballs. In *Comp. Graph. Forum*, vol. 16, C357–C364.

O'BRIEN, J., BARGTEIL, A., AND HODGINS, J. 2002. Graphical modeling and animation of ductile fracute. *ACM Trans. on Graph. 21*, 3, 291–294.

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ, P., GROSS, M., AND GUIBAS, L. 2005. Meshless animation of fracturing solids. In *ACM Trans. on Graph.*, vol. 24, 957–964.

PLA-CASTELLS, M., GARCÍA-FERNÁNDEZ, I., AND MARTÍNEZ, R. 2006. Interactive terrain simulation and force distribution models in sand piles. *Cellular Automata*, 392–401.

SELLE, A., LENTINE, M., AND FEDKIW, R. 2008. A mass spring model for hair simulation. In *ACM Trans. on Graph.*, vol. 27, 64.1–64.11.

ST LAWRENCE, W., AND BRADLEY, C. 1975. The deformation of snow in terms of structural mechanism. In *Snow Mech. Symp.*, 155.

STEFFEN, M., KIRBY, R., AND BERZINS, M. 2008. Analysis and reduction of quadrature errors in the material point method (MPM). *Int. J. Numer. Meth. Engng 76*, 6, 922–948.

STOMAKHIN, A., HOWES, R., SCHROEDER, C., AND TERAN, J. 2012. Energetically consistent invertible elasticity. In *Eurographics/ACM SIGGRAPH Symp. on Comp. Anim.*, 25–32.

SULSKY, D., ZHOU, S.-J., AND SCHREYER, H. 1995. Application of particle-in-cell method to solid mechanics. *Comp. Phys. Comm. 87*, 236–252.

SUMNER, R., O'BRIEN, J., AND HODGINS, J. 1999. Animating sand, mud, and snow. In *Comp. Graph. Forum*, vol. 18, 17–26.

TERZOPOULOS, D., AND FLEISCHER, W. 1988. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Proc. ACM SIGGRAPH 1988 22*, 4, 269–278.

WISCOMBE, W., AND WARREN, S. 1980. A model for the spectral albedo of snow. I: Pure snow. *J. of the Atmospheric Sciences 37*, 12, 2712–2733.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. In *ACM Trans. on Graph.*, vol. 27, 47.

YU, J., AND TURK, G. 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 217–225.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. on Graph. 24*, 3, 965–972.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM Trans. on Graph.*, vol. 24, 965–972.

ZHU, B., AND YANG, X. 2010. Animating sand as a surface flow. *Eurographics 2010, Short Papers*.

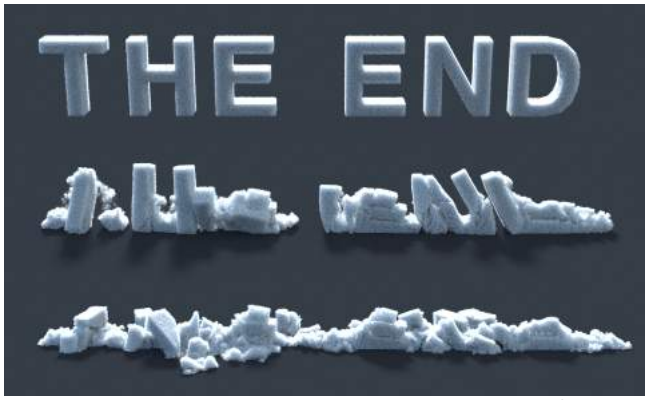

Figure 12: The end. *We can simulate other words, too.* ©*Disney.*