Adaptive Merging for Rigid Body Simulation

EULALIE COEVOET, OTMAN BENCHEKROUN, and PAUL G. KRY, McGill University

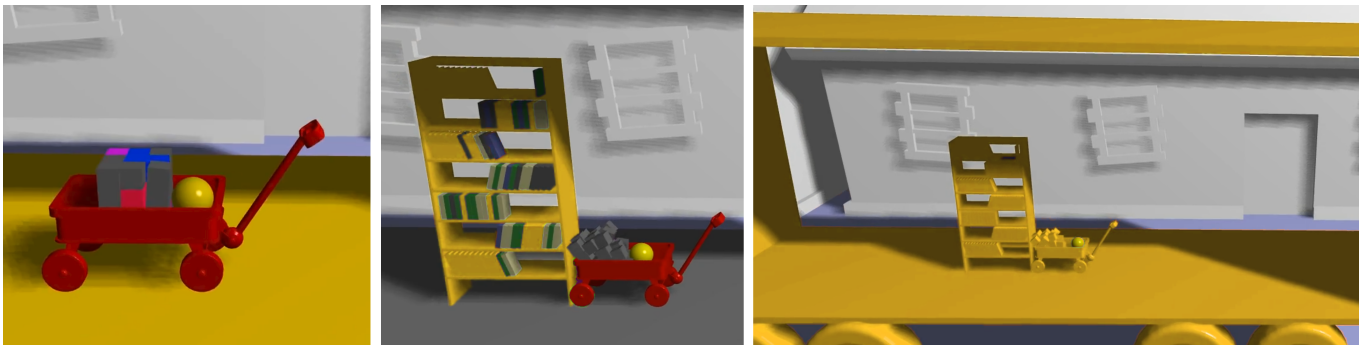

Fig. 1. Simulation of a moving wagon filled with toys, colliding with a bookcase full of books, while being transported in the back of a moving truck. Collections of bodies with common relative velocity merge and simulate as one, displayed here in different bright colors. Only a fraction of the bodies in the scene needs to be simulated at any given time step, with bodies merging into and out of collections as necessary due to external forces, and interactions (e.g., when the wagon collides with the bookcase).

We reduce computation time in rigid body simulations by merging collections of bodies when they share a common spatial velocity. Merging relies on monitoring the state of contacts, and a metric that compares the relative linear and angular motion of bodies based on their sizes. Unmerging relies on an inexpensive single iteration projected Gauss-Seidel sweep over contacts between merged bodies, which lets us update internal contact forces over time, and use the same metrics as merging to identify when bodies should unmerge. Furthermore we use a contact ordering for graph traversal refinement of the internal contact forces in collections, which helps to correctly identify all the bodies that must unmerge when there are impacts. The general concept of merging is similar to the common technique of sleeping and waking rigid bodies in the inertial frame, and we exploit this too, but our merging is in moving frames, and unmerging takes place at contacts between bodies rather than at the level of bodies themselves. We discuss the previous relative motion metrics in comparison to ours, and evaluate our method on a variety of scenarios.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: merging, sleeping, constraints, contact, friction

ACM Reference Format:
Eulalie Coevoet, Otman Benchekroun, and Paul G. Kry. 2020. Adaptive Merging for Rigid Body Simulation. 1, 1 (October 2020), 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' address: Eulalie Coevoet, eulalie.coevoet@gmail.com; Otman Benchekroun, otmanbench@gmail.com; Paul G. Kry, kry@cs.mcgill.ca, McGill University.

XXXX-XXXX/2020/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In a multi-body system with contacts, the motion of each body is dependant on external forces and the contacts that exist between bodies. Determining where the contacts occur and the forces they produce are the two most computationally expensive steps of a rigid-body simulation. Methods to speed-up these computations while maintaining plausible (or correct) motion are valuable as it allows the complexity of the scene to be increased, and this is of particular interest in the context of interactive simulations.

Sleeping and waking is a popular mechanism in most physics engines to reduce the compute time by not performing collision detection between sleeping bodies, and avoiding the calculation of their resting contact forces. A sleeping body, and typically the neighboring bodies in contact, will wake upon receiving an external force, making it still possible for a user to freely interact with all bodies, sleeping or otherwise, in a large simulated world. This can be very effective for eliminating unnecessary computation. However, sleeping alone misses an important opportunity: allowing large groups of bodies to sleep relative to one another, even if they are moving in the inertial frame. We call this *adaptive merging*, and it can arise, for instance, when a vertical stack of books is pushed at its bottom-most book; the merged collection of bodies can slide together as one merged object on a flat surface in the simulation. Merging can also occur in scenarios where bodies share a common rotational velocity, for example, with objects on a swing, or stacks of equipment on a rotating space station. Figure 1 shows an elaborate scenario involving many books and toys on different moving platforms, which we simulate efficiently, with collections of objects unmerging only when necessary.

Merging allows us to bypass collision detection for bodies that are members of the same collection. Contact force computations are likewise only necessary between collections and active bodies in the scene. To make this all possible, and to allow bodies to unmerge,

we use a single iteration of an iterative projected Gauss-Seidel (PGS) solve at each time step to maintain an approximation of the contact forces internal to the collection. Thus, when we observe the internal contact forces between merged bodies reduce to zero, or become clamped to the friction cone, we know that the body is ready to unmerge from its neighbor in the contact graph. To ensure the timeliness of unmerging events, we use a careful ordering of the contact graph in our single iteration PGS refinement of internal forces. Furthermore, while we discard the velocity update produced by the single iteration solve, we still use it to monitor the approximately desired velocities of bodies within a collection, with respect to the collection, to know if they should unmerge. Thus our umerging and merging criteria have symmetry, which aids in setting thresholds.

Finally, an interesting challenge that we address in this work is the identification of good metrics to identify when bodies can merge. The energy-like metrics used in previous sleeping work are not appropriate for merging bodies into non-stationary collections, and can involve potentially tricky choices for weighting the angular and linear components of relative rigid body velocities. We present criteria that are based on a linear speed limit, for instance, the speed of a sliding contact, or velocity of a point on a body with respect to the motion of both. Thus, setting a threshold is straightforward based on the objects in the scene. Setting a speed limit of 0.1 mm per second is an easy choice for parking a car, for instance, on the sloped ramp of a ferry, and letting it merge with the boat. With merging there is an additional stabilization benefit. Frictional contact solutions do not typically achieve perfectly zero relative velocity in these scenarios. After a voyage at sea, without adaptive merging, we might discover that the car is lost at sea, having slipped off the boat despite a sufficiently high coefficient of friction.

Our techniques can be added on top of any variety of 2D or 3D multi-body simulation engines, regardless if they use iterative or direct solvers for their main contact constraint solve. We are also agnostic to the choice of friction cone (exact, pyramid, or box) as long as we can approximate the appropriate frictional contact behaviour within a projected Gauss-Seidel like solver so as to monitor forces between bodies inside collections. Our method reveals significant speed-ups while maintaining plausibility in our example scenes. Important features of our contribution include,

- the merging of bodies with a common relative velocity;
- a single iteration PGS to refine forces internal to collections;
- contact ordering for responsive unmerging; and
- easily tuned parameters for merging and unmerging.

2 RELATED WORK

Adaptive simulation has been an important research area across all aspects of physics-based animation. Manteaux et al. [2017] provides a survey of the many recent efforts in adaptive models for simulation. Deformation has received the lions share of the focus given the great potential for fine visual details to be produced with vastly reduced compute times, for elastic solids [Tournier et al. 2014], cloth [Narain et al. 2012], and fluids [Ando et al. 2013], as several examples. For rigid bodies, in contrast, the problem is different because we are not avoiding the potentially unlimited dimension of the state vector in a finely discretized deformable continuum. Nevertheless, there still exists excellent opportunities for adaptive simulation to improve performance in scenarios with large numbers of bodies and contacts.

Sleeping and Adaptive Approaches. Most physics engines have some form of sleeping body feature. Bullet, Physx, and Vortex all have sleeping in the inertial frame, and the academic work on this topic is now well established and appreciated for the impact it has had on the games community. The PhD work of Schmidl [2002] proposes a heuristic for freezing bodies based on kinetic energy, as a means of reducing computation time in rigid body simulations. Schmidl and Milenkovic [2004] and Erleben [2004] likewise discuss body freezing strategies for saving computation. They monitor energy (with some mass weighting) over a time window, and then freeze bodies that they determine to be stationary in the inertial frame.

Guendelman et al. [2003] use a variation of these freezing methods to handle shock propagation in the stacking of non-convex rigid bodies. Also in the context of stacking rigid bodies into messy piles, Hsu and Keyser [2010] propose an optimization that captures the overall pile behaviour without computing all the frictional contact interaction between individual bodies.

Redon et al. [2005] present an adaptive approach to freezing and unfreezing joints in an articulated multi body system. This is one of the few techniques that allows merging bodies into a *moving* common frame that is not the inertial frame. It is the individual joints that become rigid when joint accelerations are low, effectively simplifying the articulated body system. Müller et al. [2017] generate long range constraints to improve convergence of iterative solves, for piles of bodies, and for articulated mechanisms with joint limits. This employs ideas similar to long range attachments previously proposed for cloth simulation [Kim et al. 2012], and while not adaptively reducing complexity (i.e., this approach selectively *increases* the number of constraints in the system), it is very effective at speeding up iterative solves that require force information to propagate across a large numbers of bodies.

Another adaptive approach of interest is that of Artemova and Redon [2012], which proposes an adaptive Hamiltonian for freezing particles in the inertial frame. Momentum accumulates at frozen particles, and these particles can then thaw once they have gained sufficient momentum. The technique has been applied to fluids and cloth [Manteaux et al. 2013], and in general provides some inspiration for the approach we take in this paper; instead of updating momentum, we update contact forces internal to collections to know when they should unmerge.

Collision Detection. While the focus of our work is to reduce collision processing cost, collision detection is an important component for any rigid body simulator. The concern is not only the computational cost of collision detection, but also the contacts that it produces because this can have an important impact on the quality of a simulation [Erleben 2018]. Broad and narrow phases that include bounding volume hierarchies (BVHs) and other spatial data structures are typical [Ericson 2004]. For meshes we use sphere trees constructed in a manner similar to that of Quinlan [1994] but based on tight medial sphere representations [Stolpner et al. 2012]. When detection provides too many contacts between a pair of bodies, contact pruning is an important step that we employ in our 2D tests

(e.g., identifying and eliminating redundant contacts). Furthermore, with collision processing time being proportional to the number of constraints generated, we note that four sided discrete friction cones appear to be the most popular for LCP formulations of contact, to minimize the number of constraint rows. Finally, while merged collections remove the need to check bodies within the same collection, our approach can still benefit from a variety of other techniques, such as fast BVH reconstruction [Lauterbach et al. 2009].

Acceleration Techniques. Other techniques for speeding up rigid body simulation include speculative integration, using the time warp algorithm as proposed by Mirtich [2000]. Parallelism can be likewise be exploited, for instance by splitting connected components of the contact graph into islands, as described by Parker and O'Brien [2009] in the context of fracture simulation. Parallelism can also be easily exploited with iterative methods, such as block-Jacobi variants of projected Gauss-Seidel for rigid bodies [Tonge et al. 2012], and position based dynamics [Fratarcangeli and Pellacini 2015]. For direct solvers, parallelism is also possible with Schur complement substructuring [Peiret et al. 2019]. While many approaches speed up the computation at a given time step, our strategy, in contrast, is to merge bodies into a collection where they can require minimal computation over a large number of time steps.

3 MULTIBODY DYNAMICS

Bender et al. [2013] provide an excellent review of rigid body simulations, including equations of motion, frictional contact, joint constraints, as well as different formulations, solvers, and parallelization. Here we briefly review the equations and notation that we use in this paper, and while some later examples are given in 2D for clarity, we will focus on the slightly more complex 3D case.

The dynamic equations of motion for a multi-body system with contacts is given by,

$$\mathbf{M}\dot{\mathbf{u}} = \mathbf{f} + \mathbf{c}(\mathbf{u}) + \mathbf{J}^T\lambda. \tag{1}$$

Here, $\mathbf{M}$ is the generalized mass, a block diagonal matrix with the mass of each rigid body. We follow the common practice of using a world aligned frame at the center of mass of each body to express the linear and rotational body velocities, thus, $\mathbf{M}$ contains rotational inertia sub-matrices that are rotated into the world aligned frame given the body's current orientation. The generalized velocity $\mathbf{u}$ concatenates the velocity $\mathbf{u}_b = (\mathbf{v}_b^T, \boldsymbol{\omega}_b^T)^T$, linear then angular, for each body b, in coordinates aligned with the world frame. Finally, we have $\mathbf{f}$ combining external and velocity dependent forces, and the contact Jacobian providing forcing directions $\mathbf{J}^T$, with normal and friction force magnitudes in Lagrange multipliers λ .

Following the formulation of Erleben [2007], we first solve for velocities and a velocity correction due to contact, and then advance positions implicitly with these velocities, using Rodrigues' formula for the rotation update. We use $\mathbf{R}_b$ to denote the orientation and $\mathbf{x}_b$ the center of mass position of body b with respect to the world frame.

Frictional Contact Solve. The constraint-based solve of contact forces in the normal direction is subject to the Signorini condition, yielding the complementarity

$$0 \leq \mathbf{w}_n \perp \lambda_n \geq 0, \tag{2}$$

where $\mathbf{w}_n = \mathbf{J}_n\mathbf{u}$ is the relative velocity between two colliding objects in the normal direction at the frame of the contact point. This condition ensures a physical response with no interpenetration, and that the separation between the two objects at a contact point is zero when its contact force is applied.

We use the Coulomb friction model, and include two tangent forcing directions in the Jacobian. Thus, the contact forces are

$$\mathbf{J}^T\lambda = \begin{bmatrix} \mathbf{J}_n^T & \mathbf{J}_{t1}^T & \mathbf{J}_{t2}^T \end{bmatrix} \begin{bmatrix} \lambda_n \\ \lambda_{t1} \\ \lambda_{t2} \end{bmatrix}. \tag{3}$$

As opposed to defining bounds on λ for the exact friction cone, in this work we use a four sided pyramid discretization of the friction cone. Tangential and normal forces are then linked by the following constraints. For a contact i, the normal direction

$$(\mathbf{w}_n)_i \geq 0, (\lambda_n)_i \geq 0, \quad \text{and} \quad (\mathbf{w}_n)_i(\lambda_n)_i = 0 \tag{4}$$

and in the tangential direction $t1$,

$$(\mathbf{w}_{t1})_i > 0 \quad \Rightarrow \quad (\lambda_{t1})_i = -\mu(\lambda_n)_i \tag{5}$$

$$(\mathbf{w}_{t1})_i < 0 \quad \Rightarrow \quad (\lambda_{t1})_i = \mu(\lambda_n)_i \tag{6}$$

$$(\mathbf{w}_{t1})_i = 0 \quad \Rightarrow \quad |(\lambda_{t1})_i| \leq \mu(\lambda_n)_i \tag{7}$$

Here, μ is the coefficient of friction, and $\mathbf{w}_{t1} = \mathbf{J}_{t1}\mathbf{u}$ is the relative velocity between two colliding objects at the contact point in direction $t1$, or slip velocity. The same constraints apply for $\mathbf{w}_{t2} = \mathbf{J}_{t2}\mathbf{u}$ and λ_{t2}.

We use an explicit Euler integration scheme to integrate velocities over time with Equation 1. Given a time step h and a current velocity state $\mathbf{u}_-$, the velocity state $\mathbf{u}_+$ at the end of the time step is given by

$$\mathbf{u}_+ = \mathbf{u}_- + h\mathbf{M}^{-1}\mathbf{f} + h\mathbf{M}^{-1}\mathbf{J}^T\lambda. \tag{8}$$

Thus, $\mathbf{w}$ is then given by,

$$\mathbf{w} = \mathbf{J}\left(\mathbf{u}_- + h\mathbf{M}^{-1}\mathbf{f}\right) + h\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda. \tag{9}$$

We can use a variety of methods to solve the contact constraint forces in Equation 9, including both direct and indirect solvers. In our work, we use a PGS solve for the simplicity of being able to use it for both our single iteration update as well as the full LCP solve. In contrast, we note that the high accuracy of direct solvers would be beneficial for a faster identification of merging events.

4 RIGID BODY COLLECTIONS

When the relative motion between two bodies is low for a period of time, the bodies can be *merged* giving form to a *collection* of bodies moving together. Intuitively, this acts like a temporary weld, and is only made between two bodies in contact when they satisfy criteria we will discuss in later sections. Here we describe the process of computing the properties of a collection, and once formed we effectively treat the collection as a single body in our simulation framework. Note that when bodies are part of a collection, we call them *merged* bodies. In contrast, when bodies exist in the simulation as individuals, we call them *free* bodies.

When we merge a body b with a collection c, the new body influences the collection's total mass m_c, rotational inertia $\mathbf{I}_c$ and center of mass $\mathbf{x}_c$. With a change to the center of mass, we must also update the linear velocity $\mathbf{v}_c$ based on the angular velocity $\boldsymbol{\omega}_c$.

We start by computing the common center of mass as a simple weighted average,

$$\mathbf{x}_{\text{com}} = \frac{m_c\mathbf{x}_c + m_b\mathbf{x}_b}{m_c + m_b}.$$

We then compute the rotational inertia of the collection relative to the new center of mass. The collection and the body will both contribute their current rotational inertia, expressed in the world aligned frame, $\mathbf{I}_c$ and $\mathbf{I}_b$, respectively. But each body will also contribute an additional rotational inertia, due to its mass and the displacement of its center of mass from the new center of mass. Thus,

$$\mathbf{I}_{\text{com}} \leftarrow \mathbf{I}_c + m_c\hat{\mathbf{r}}_c^T\hat{\mathbf{r}}_c + \mathbf{I}_b + m_b\hat{\mathbf{r}}_b^T\hat{\mathbf{r}}_b$$

where $\mathbf{r}_c = \mathbf{x}_c - \mathbf{x}_{\text{com}}$, and $\mathbf{r}_b = \mathbf{x}_b - \mathbf{x}_{\text{com}}$ and where $\hat{\mathbf{r}}$ is the 3-by-3 cross product matrix $\mathbf{r}\times$ (see, for instance [Lee and Goswami 2007]). Given the current orientation of the collection $\mathbf{R}_c$, we also update the rest pose angular inertia $\mathbf{I}_{c0} = \mathbf{R}_c^T\mathbf{I}_{\text{com}}\mathbf{R}_c$ and the rest pose inverse inertia as it is needed for solving the equations of motion.

For the merge to take place, the relative velocity between the bodies will be very low. Thus they effectively share a common spatial velocity (expressed in different coordinate frames due to different center of mass positions). However, the spatial velocities can still differ a small amount based on the merging threshold. Thus, we carefully set first the new angular velocity and then the new linear velocity while ensuring a conservation of momentum:

$$\boldsymbol{\omega}_c \leftarrow \mathbf{I}_{\text{com}}^{-1}\left(\mathbf{I}_c\boldsymbol{\omega}_c + m_c\hat{\mathbf{r}}_c\mathbf{v}_c + \mathbf{I}_b\boldsymbol{\omega}_b + m_b\hat{\mathbf{r}}_b\mathbf{v}_b\right), \tag{10}$$

$$\mathbf{v}_c \leftarrow \frac{m_c\mathbf{v}_c + m_b\mathbf{v}_b}{m_c + m_b}. \tag{11}$$

Finally, we update the mass, rotational inertia, and position of the collection,

$$m_c \leftarrow m_c + m_b, \tag{12}$$

$$\mathbf{I}_c \leftarrow \mathbf{I}_{\text{com}}, \tag{13}$$

$$\mathbf{x}_c \leftarrow \mathbf{x}_{\text{com}}. \tag{14}$$

The process is the same, regardless if it is two free bodies forming a collection, or the merging of two collections. We also compute these updates in reverse to remove a body from a collection, or likewise to remove a collection of bodies all at once.

The only special case is if one of the bodies is pinned. In this case, we treat the mass as infinite, and the collection velocity assumes the velocity of the pinned body (i.e., zero). Furthermore, the center of mass position is irrelevant, and we use zero for inverse mass and inverse inertia in all our computations (e.g., resolution of contact forces).

Note that when a collection is formed or updated, we record the relative position of each merged body with respect to the collection. This transformation is needed for drawing the body, for computations related to our single iteration PGS refinement of internal contact forces, and for when it unmerges. We also update an approximate oriented bounding box (OBB) for the collection. We use the OBB in our relative motion merging criteria (see Section5.2).

In the rest of the paper, we let $\mathcal{B}$ be the set of rigid bodies being *simulated*, meaning that bodies in $\mathcal{B}$ are checked for collision detection and are part of the LCP system (i.e., merged bodies are not part of $\mathcal{B}$ as individuals; $\mathcal{B}$ only contains the free bodies and collections).

4.1 Collision Detection for Collections

While the main advantage of a collection is that it reduces collision processing cost, collision detection also has an important impact on the total compute time. A minimal collision culling approach for collections would be to check a single bounding volume, and if there is overlap, then proceed to test with all bodies inside the collection, regardless of its size. Given that some collections can be large *and* long lived, it can be very valuable to build hierarchies on the fly. Given that we already employ a sphere based BVH collision detection acceleration for some mesh objects, we extend our implementation to also work with collections. We update the hierarchy in an incremental manner whenever a body is added to or removed from the collection, and use a naive tree balancing approach.

5 MERGING

First we define criteria to decide when it is reasonable to merge bodies together. For sleeping policies (i.e., merging with the inertial frame), a common approach is to look at either the velocities, or the kinetic energy of the body in question. Schmidl [2002] keeps track of each body's kinetic energy over multiple time steps, and checks that this value is below a threshold. They use the momentum a body picks up during a time step, as a result from gravity, to define the threshold. Erleben [2004] adds the condition that the kinetic energy should be strictly non-increasing, and proposes to scale an arbitrary threshold given by the user with the mass of the body. This allows massive bodies to fall asleep as easily as light ones. Using kinetic energy is a good strategy to deal with only one threshold. Schmidl [2002], Millington [2007], and Lengyel [2011] all watch the velocities, using two different thresholds instead: one for linear velocity and one for angular velocity. A temporal condition is used in each of these works, avoiding to put at rest bodies that experience a fleeting zero velocity (e.g., a ball at the peak of its trajectory when thrown straight up in the air).

For merging (not only to the inertial frame), one approach is to look at the relative motion. Redon et al. [2005] use an acceleration metric, and deactivate joints of articulated rigid bodies if the metric is below a threshold given by the user, which is equivalent to merging. We propose a similar strategy by looking at the relative velocity, and in addition, because we are in the context of multi-body simulation, we also look at the contact forces. We design our merging and unmerging criteria in symmetry, and because contact forces are going to be the central key of the unmerging strategy, we propose the use of these forces in the early stage of merging.

5.1 Insights on Merging Criteria

We allow two rigid bodies of $\mathcal{B}$ to merge, if they are in contact and satisfy each of the two following criteria:

- **Relative motion**: For a given number of time steps, the relative velocity between the two objects should stay below a given threshold. We use relative velocity rather than energy

(i.e., as measured with effective mass and relative velocity) because we want massive objects to merge as easily as light ones.

- **Contact state**: If the contacts are on the edge of their friction cone, it is likely that the bodies are not in a stable position. Therefore, we check for sliding contacts between the two objects and merge only if the configuration is considered as being stable for the same number of time steps used in our relative motion criterion.

Note that the last condition is used to prevent merging two bodies that are likely to move in the near future (with respect to each other). However, it is acceptable to use the relative motion metric on its own to decide when to merge. We give more details on each criterion in the following sections.

5.2 Relative Motion Metric

The relative motion metric is computed after solving the LCP system for each pair of bodies, $b_1, b_2 \in \mathcal{B}$, that are in contact. We propose to compare the threshold, set by the user, to the fastest point of the two bodies relative to each other. To that end, we use the bounding box of each body, compute the relative spatial velocity at each point of the two oriented bounding boxes, and select the highest:

$$v_{\text{rel}}^{\max} = \max_{\mathbf{p} \in \text{OBBs}} |\mathbf{v}_{\text{rel}}(\mathbf{p})|, \tag{15}$$

with

$$\mathbf{v}_{\text{rel}}(\mathbf{p}) = \dot{\mathbf{x}}_{b1} + \boldsymbol{\omega}_{b1} \times (\mathbf{p} - \mathbf{x}_{b1}) - \dot{\mathbf{x}}_{b2} - \boldsymbol{\omega}_{b2} \times (\mathbf{p} - \mathbf{x}_{b2}). \tag{16}$$

A good example to justify this approach is a see-saw configuration (see Figure 2). The relative velocity computed in a frame at the common center of mass (which corresponds to the point of rotation) can seem very small and fall below a naive threshold, while a slow relative motion can be more easily observed at the extremities of the see-saw. Using the above computation, we prevent from merging in such a case.

With $v_{\text{rel}}^{\max}$ the fastest relative linear velocity between the two bodies, the *relative motion metric* condition is then given by

$$v_{\text{rel}}^{\max} < \epsilon, \tag{17}$$

with ϵ the threshold provided by the user. This metric is computed from the new velocities (i.e., results of the LCP solve) and the current position of the bounding boxes. Note that we can similarly merge a body to the inertial frame by computing its highest velocity using the same bounding box method. This happens when we merge with pinned bodies, in which case we only consider the OBB of the unpinned body. We will give more details on merging with the inertial frame in Section 7.

5.3 Monitoring Contact State

In combination with the relative motion metric, we suggest to monitor the contact state to decide if a merging is *necessary*. Indeed, even with a low relative velocity, if we detect that contacts between the two bodies are likely to slide in an immediate future, we could avoid a merge that will probably not last into the next time step.

With friction involved, a contact has three possible states; inactive, active and in static friction, or active and in dynamic friction (i.e., sliding). At this stage we are only looking at active contacts, i.e., detected contacts that have non-zero Lagrange multipliers after the LCP solve. It is the transition between static and dynamic friction that is of importance here, and this is detected by checking if the contact force is on the edge of the friction cone, that is, $|(\lambda_{t1})_i| = \mu(\lambda_n)_i$ for a contact i, and similarly for $t2$. But it can be more reliable to monitor sliding motion with an LCP solution that has not fully converged. Thus, we only check if the slip velocity exceeds an upper bound, that is,

$$(\mathbf{w}_{t1})_i < \epsilon, \tag{18}$$

for each contact i. A similar bound is applied to $\mathbf{w}_{t2}$. We do not merge the pair of bodies if the limit has been reached.

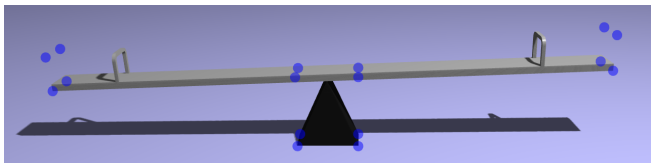

Fig. 2. See-saw configuration. The blue spheres are the corners of the oriented bounding boxes used to compute the point of fastest relative velocity for the two bodies.

Let it breathe. Certainly our PGS solve of frictional contact forces does not give us a minimum norm solution. To limit such bothersome results in our monitoring of the contact state, we use a well known method which consists of adding compliance and Baumgarte stabilization (i.e., feedback) to the LCP [Baumgarte 1972]. Intuitively, it regularizes the problem, and it is similar to surrounding rigid bodies with a very thin deformable layer, and allowing some interpenetration (constraint violations). If we picture a stack of bodies dropped on the floor, this compliance and feedback interaction will look like the stack is *breathing* after it hits the floor, as each body moves slightly to minimize the constraint violations in the system. We avoid merging bodies while the Baumgarte feedback is working to stabilize the constraint violation and the system is *breathing*. To that end, we monitor the change in constraint violations (i.e., the interpenetration change) between time steps, and wait for it to be lower than a given threshold.

6 UNMERGING

The main challenge of sleeping and merging techniques is to design a reactivation strategy that ensures plausible motion. Different strategies have been used for waking bodies. Hsu and Keyser [2010] use a momentum heuristic, i.e., if an object with sufficiently high momentum hits a pile of sleeping bodies, the pile is entirely woken to solve the force propagation. They combine this check with what they call an avalanche strategy, which triggers when the upper levels of a pile cannot be supported by those underneath. Schmidl [2002] wake a body if it picks up momentum through a collision. One common strategy is to propagate the reactivation to all the direct neighbours of the body and a given number of indirect neighbours. Another technique mentioned by Manteaux et al. [2017] is to keep the bodies asleep for a specified duration and wake them up when that duration ends.

In our case, an unmerge can be triggered by more than just a collision. Consider a pile of merged bodies moving together with a platform. The motion of the platform alone can cause some bodies to unmerge as it starts to tip over. Redon et al. [2005] reactivate the joints of an articulated rigid system, if their corresponding acceleration metric is higher than a given threshold. The key concept of their technique is that this metric can be computed prior to the acceleration of the joints themselves. Our goal is to propose a strategy that produces plausible dynamics (e.g., response to collisions, gravitational forces, springs, and user interactions) while not being computationally expensive. We think a good approach is to keep track of the internal contact forces that the bodies experience inside the collection.

6.1 Insights on Unmerging Criteria

With the desire to have symmetry between merging and unmerging, we unmerge when the following criteria are encountered:

- **Contact state**: Within the collection, if a contact i between two bodies breaks, i.e., $(\lambda_n)_i = 0$, or was detected as being about to slide during the time step, we remove the weld between the pair of bodies. If this separation results in breaking the collection's contact graph into separate components, we unmerge.
- **Relative motion**: For relative motion, we look at a desired velocity update from a single iteration PGS solve. If the relative motion metric between the two bodies is higher than a given threshold for a given number of time steps, and if the number of contacts between the two bodies is smaller than three, the weld between the two bodies is also removed. Note that counting the contacts is only pertinent with an efficient pruning approach.

6.2 Updating collections and contact ordering

When we merge two bodies of $\mathcal{B}$, contact force information between them is saved to build a model of what constitutes the range of contact forces experienced while observing no relative motion (i.e., a range of plausible forces one might expect within a merged set of bodies, even though we are not simulating them). At each time step of the simulation, while the collection's position and velocities are updated, if we do nothing else, then the saved contact forces within the collection will no longer be accurate (e.g., when adding a body on top of a pile). Yet, these contact forces give valuable information for deciding when to unmerge. We propose to approximate the correct forces by computing a single iteration of the PGS algorithm, a single sweep, at each time step of the simulation. During this sweep, all the bodies are included (both *merged* and *free* bodies).

To guarantee that this cheap resolution provides profitable results, we use the contact forces (stored from the previous time step) as a warm start for the PGS algorithm. Note that when large changes occur during a time step (e.g., collision with another object or a large velocity change), contact forces are more likely to be significantly over or under estimated. However, these contact forces will converge over time.

PGS updates constraints one at a time, assuming other variables are fixed, with a projection step to enforce the complementarity constraints. While the solution can converge quickly, a single iteration is not necessarily a good approximation for the true solution, even with a good warm start. Thus we take special care with the order in which each contact is processed because it can have a large influence on the results.

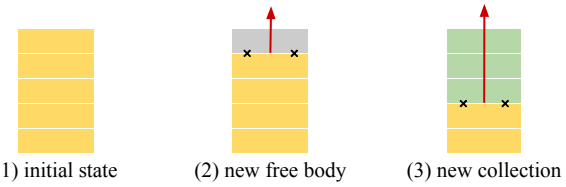

Fig. 3. Unmerging illustration. The merged bodies are displayed by color, the free bodies are in dark grey. (2) A breaking contact is detected, setting free one body of the collection. (3) A breaking contact is detected, separating the collection into two parts.

To obtain a good approximation of the contact forces in the collections, we propose to order the contacts, starting with those that have been newly detected at the current time step (e.g., impacts). We then walk the constraint graph starting from these locations in a simultaneous breadth first manner. This gives a good opportunity for the response at new contacts to be propagated through the collection. We proceed the same way with bodies interactively moved by the user. If the body has known contact points, then locations in the constraint graph will always come first.

6.3 Monitoring Contact State for Unmerging

Once the contact forces have been updated, we look at the forces and check for broken or sliding contacts. If a contact breaks during this update, we remove the weld between the pair of bodies. Note that, if a contact is inactive when two bodies are merged, we discard it from the contact graph, avoiding the immediate triggering of an unmerge event. For the detection of sliding contacts, we use the same approach as described in Section 5.3. Because the slip velocities $\mathbf{w}_{t1}$ and $\mathbf{w}_{t2}$ available from the one iteration PGS can be cheap approximations, we combine the test with a test of the Lagrange multipliers. We tag contact i as *on edge* if

$$(\mathbf{w}_{t1})_i > \epsilon \tag{19}$$

and

$$|(\lambda_{t1})_i| = \mu(\lambda_n)_i. \tag{20}$$

We determine this slip similarly for $\mathbf{w}_{t2}$, and λ_{t2}. For each collection, we loop over the body pairs and collect the broken welds. The removal of the welds will give a new contact graph for the collection. The unmerge step then consists of a traversal of this new graph to obtain the resulting free bodies and new collections, if any exist (see Figure 3). If the welds removal leaves the collection unchanged, then nothing will be unmerged. It is important to perform this step between the one iteration PGS solve and the full LCP solve. This ensures a plausible response when a body collides with a collection, or when the user interacts with a collection. Likewise, it allows us to get accurate contact forces and velocities for the newly unmerged bodies. Note that each unmerged body has the spatial velocity of its collection at the moment it is unmerged.

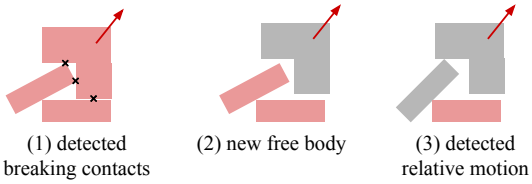

Fig. 4. Unmerging illustration. The merged bodies are displayed by color, the free bodies are in dark grey. (1) Breaking contacts are detected. (2) One body is set free. (3) A contact acts as a pivot which is detected by the relative motion check.

6.4 Unmerging with the Relative Motion Metric

While contact forces provide, in most cases, enough information to decide when to unmerge, relative motion can be observed while nothing triggers the contact state condition. These events correspond to contacts acting as pivot (see Figure 4), and this configuration is only possible if there are only one or two contacts between a pair of bodies. To detect such cases, we use the same *relative motion metric* that we described for merging.

The velocities used to compute the metric are the results of the single iteration PGS solve. Because these velocities are cheap approximations, we also track the metric over multiple time steps. If there are less than three contacts between the pair of bodies, and if the metric is greater than the threshold, we start to store its value. If the metric stays greater than the threshold for a given number of time steps, the weld is removed. If the metric is smaller than the threshold for a single time step, the accumulation (storage) is canceled and the count restarts from zero. Note that this strategy requires the collision detection to provide reasonable contact information. If no efficient pruning strategy is available, the same approach can be used without counting the contacts, though this can make unmerging more conservative (i.e., more likely).

In practice, using the same threshold as the one we use for merging is reasonable, but it can also be set higher if we want the method to be more aggressive. Note that the time step count parameter, which requires the metric to exceed the threshold for a given number of consecutive steps before unmerging, must not be set too high, otherwise we observe overly sticky results.

7 SLEEPING

Sleeping can be used on top of the merging algorithm we propose. With the same relative motion metric, each body of $\mathcal{B}$ is compared with the inertial frame. When a body is at rest for a given number a time steps, it is put to sleep and likewise removed from the system. In particular, the contacts in sleeping collections are no longer updated, which reduces the computation time even further.

The approach for waking can then be limited to a change in external forces (e.g., collision). If a new contact or a user interaction with the collection is detected at the beginning of the time step, we wake the collection. As this is done before any contact solving, when a collection wakes it is put back into the single iteration PGS solve, for contact and velocity updates. The collection is then also immediately ready for unmerging tests. While it requires a sweep with a single iteration of the PGS algorithm, the method we propose offers a more physical approach than other commonly used waking strategies. That is, we may wake a body from sleep due to a collision, but we do not unmerge it from a collection unless the PGS sweep identifies that it is going to move.

ALGORITHM 1: Compute the system state at the next time step.

```
function AdvanceTime
    ContactDetection
    ContactOrdering                          // Section 6.2
    SleepAndWake                             // Section 7
    WarmStart
    SingleIterationPGS                       // Section 6.2
    Unmerge                                  // Section 6.3 & 6.4
    SolveLCP
    AdvanceVelocities
    AccumulateRelativeMotion                 // Section 5.2
    AdvancePositions
    Merge                                    // Section 5
end
```

Table 1. Default parameters values, with all units being meters per second except for the step count parameters. These produce good behavior across all of our examples, as all scenarios have relatively similar scale.

Merging relative velocity	1e-2
Unmerging relative velocity	2e-2
Sleeping relative velocity	1e-5
Let it breath	1e-5
Sliding relative velocity	1e-2
Merging time step count	3
Sleeping time step count	3
Unmerging time step count	3

8 DISCUSSION AND RESULTS

By using this merging and unmerging approach we lower the computation expense at several levels:

(1) The size of the full LCP system is drastically reduced.
(2) No collision detection is necessary inside a collection; the contact points and forces between merged bodies are stored, and the forces are updated at each time step of the simulation.
(3) The update mentioned above is not expensive, as a single iteration of the PGS algorithm is performed. Yet, this cheap update provides an excellent indication of when to unmerge.
(4) Sleeping can easily be used on top of the merging approach, and can further reduce computation time by not performing the single iteration PGS solve for sleeping collections.

The efficiency of the method depends on the order in which each step is executed; A high level overview of the whole algorithm is given in Algorithm 1. Our method requires several thresholds to be selected for any given simulation scenario: the relative motion metric (merging and unmerging), the slip velocity, and the constraint violation change. In practice, it is not difficult to find good default

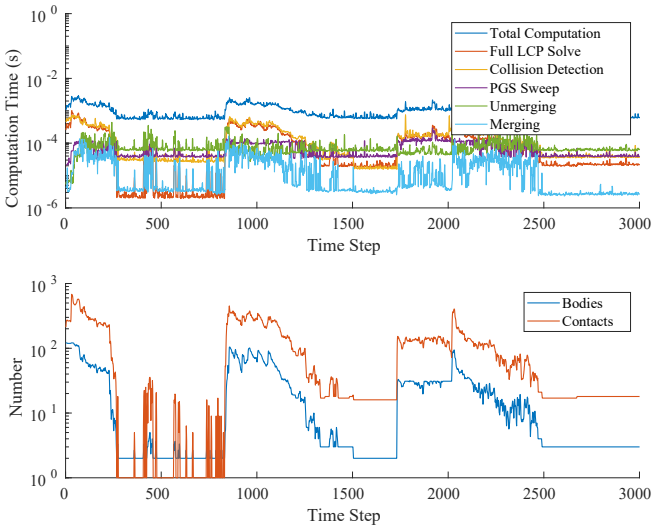

Fig. 5. Computation time and evolution of the number of bodies in $\mathcal{B}$ and contacts in the full LCP solve, for the wagon simulation.

values for pleasing results. For instance, all simulations of the Section 8.1 are run with the same default parameters of the released code (see Table 1).

8.1 Simulation Results and Applications

In this section, we present the results of our method on several challenging simulation scenarios. The computation time we give are those of the entire simulation process (i.e., collision detection, resolution, merging etc.), and are compared with our implementation of sleeping method. The full LCP is solved using the PGS algorithm with 10 iterations, which is common for interactive simulations. Note that with direct solvers (i.e., pivoting based solvers) the performance gains for big systems are more significant.

Wagon. Figure 1 shows an example involving multiple relative motions; there is a truck moving forward, a bookcase full of books at rest (with respect to the truck), and a wagon filled with toys moving at the back of the truck. We can see that some toys in the wagon are merged until the impact with the bookcase. At the time of impact some books unmerge from the bookcase and swing back and forth for a while (see the supplementary video). Finally, a short time after the impact, everything settles and merges with the truck, which has maintained a forward motion. This scenario is a good example of a dynamic simulation with lots of motion shared by groups of bodies, which once merged together, saves a lot of computation time (see Figure 5). Without adaptive merging, right after the impact between the wagon and the bookcase, the scene runs at an average of 2 ms per frame, with 118 bodies and an average of 539 contacts. With our method, the average computation time is 0.5 ms, with an average of 24 bodies and 88 contacts.

Ship crane. In Figure 6, we give an example of a training scenario for a port-crane operator. Here, simulation of a crane maneuver to load and unload containers from a ship. The figure shows stacks of

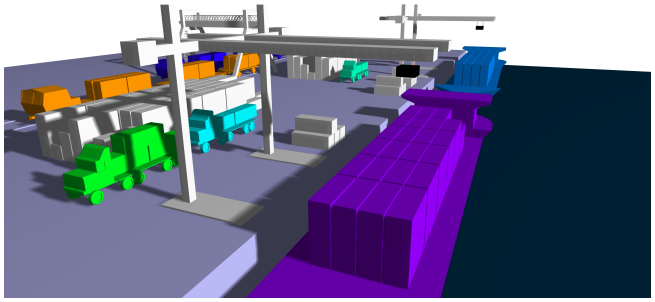

Fig. 6. Simulation of a cargo ship unload. While the containers and the crane on the platform are at rest, the ship and its cargo are moving with the sea. Collections are displayed in bright colors, and sleeping bodies in light grey.

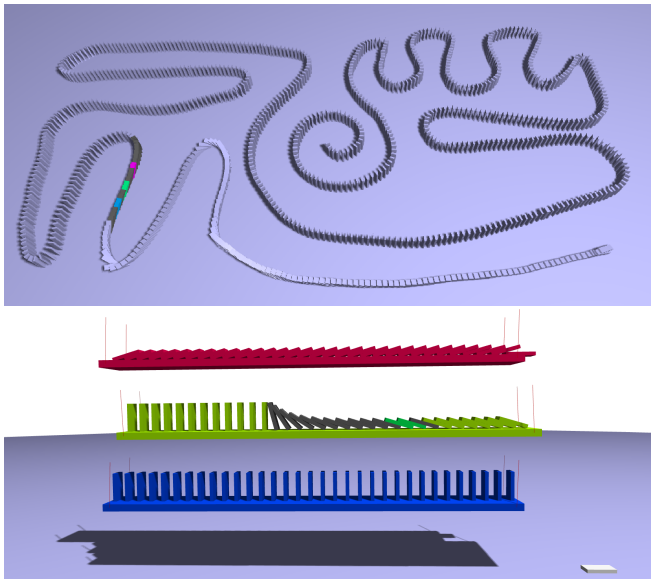

Fig. 7. Simulations of dominoes falling using our method. Free bodies are displayed in dark grey, and collections in bright colors. (top) Dominoes falling on the ground. Most bodies are merged with the ground. (bottom) Dominoes falling on slightly swinging platforms.

containers merged with a ship, which is slowly rocking on water (ship-container mass ratio is 180). The electromagnetic crane is used to unload the ship, one container at a time. Each time a body is grabbed by the crane our method detects changes in the contact forces and removes the body from its collection. The scene contains many other containers, at rest on the port, or loaded on moving trains and trucks. All the containers are available for interaction, while only a few of them are important at any given time. With only sleeping, the scene in Figure 6 runs at an average of 6 ms per frame, with 212 awake bodies and 987 contacts. With our method, the same simulation runs at an average of 1 ms per frame, with 28 bodies and 15 contacts in the full LCP solve.

Dominoes. Falling dominoes also highlight the efficiency of our method. In Figure 7 (top), dominoes at rest are merged with the ground, with only the moving dominoes comprising the free bodies in $\mathcal{B}$. Figure 7 (bottom) shows a similar example, but this time with dominoes on mobile platforms. The platforms are staggered such that the last domino on each platform falls to trigger the sequence on the next platform down. The only free bodies are the dominoes that are moving with respect to their platform. Without adaptive merging, the simulation runs at an average of 5 ms per frame, with 103 bodies and 435 contacts when the dominoes falls on the third platform. With our method, the computation time average is 2 ms per frame, with 19 bodies and 63 contacts.

Funnel. In Figure 8, cubes are continuously generated every N time steps and dropped onto a mobile funnel and on the ground (Funnel-cube mass ratio is 666). This example demonstrates how our method performs on a very large scale simulation. Without adaptive merging, at a time similar to that of the snapshot shown in Figure 8 (top), the simulation runs at an average of 62 ms per frame, with 1395 bodies and 4416 contacts. With our method, the same simulation runs at an average of 18 ms per frame, with 426 bodies and 1517 contacts in the full LCP solve. Corresponding computation time details are given in Figure 8 (bottom). On this large and unstructured simulation, our method shows a *3.5*× speedup on the total compute time, with time being saved in collision detection and collision resolution (top graph). The reason for this speedup is primarily a reduced number of bodies and contacts (bottom graph).

Unstructured stack and tower on mobile platform. In Figure 11, the method is tested with an unstructured stack. Detailed computation times are given in Figure 11 (bottom). In the attached video, we ran several simulations of random stacks, which always rapidly merge into a single collection and fall asleep. With these examples, we demonstrate the stability and robustness of the method. In Figure 12, we simulate a tower of 326 bodies on a mobile platform held by four springs. We can see the results of our method when a projectile is thrown at the tower. Unmerging can be observed at the top of the tower, which is destroyed by the impacts. Some bodies fall on the ground. Bodies on the tower settle and finally merge again, while the tower is slowly swinging with the platform. With this example we demonstrate the plausibility of our results with impacts. Without adaptive merging, when no interaction occurs with the tower, the simulation runs at an average of 8 ms per frame, with 3796 contacts in the full LCP solve. With our method, the average computation time is 1 ms, with no contact in the full LCP solve, and one body in the simulation. Note that in this example, using only 10 iterations of PGS leads to slightly inaccurate results (expansion at the top of the tower). In contrast, when using 30 PGS iterations for more accuracy, computation time goes up to 21 ms without merging, while staying at 1 ms per frame with our method. Furthermore, we observe that merging and remerging after impacts tends to occurs more quickly when using larger numbers of PGS iterations. Plots of computation time are shown Figure 12 (bottom), and a comparison of merging behavior for 10, 30, and 100 iteration PGS solves can be seen in the supplementary video.

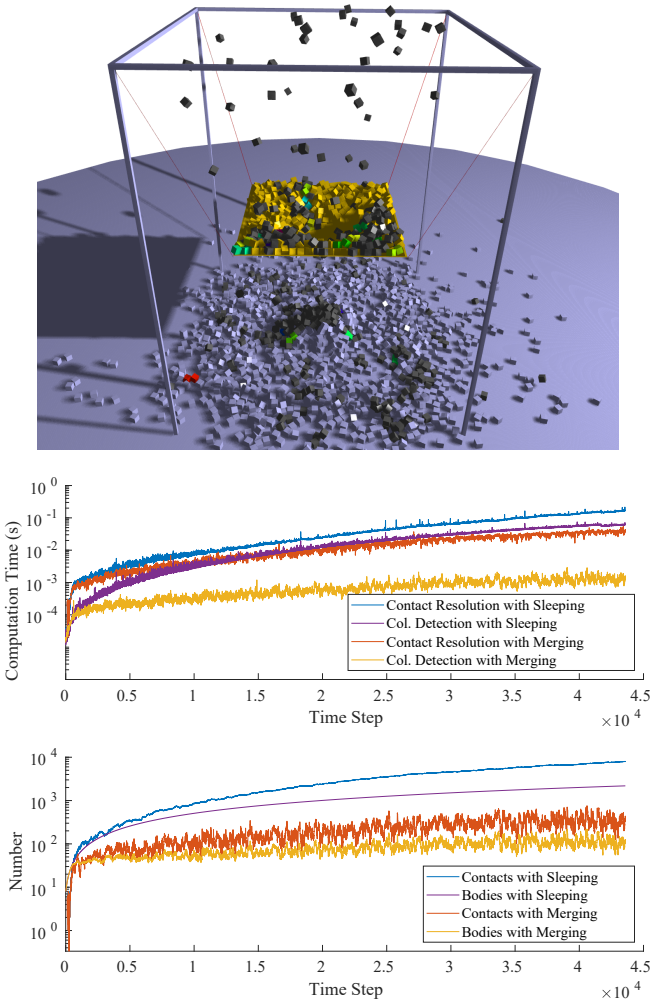

Fig. 8. Large scale simulation of cubes falling into a mobile funnel. Most bodies merge either with the funnel or the ground. Over a period of 45000 time steps, the plots compare a reference simulation with sleeping, and our method with merging, but without additional sleeping optimizations. (top graph) Contact resolution and collision detection time. With merging, the contact resolution time includes PGS sweep, merging, and unmerging computations along with the full LCP solve. The collision detection time for the merging case uses our BVH optimization for collections. (bottom graph) Number of contacts and of bodies in each case.

Spinner. We have tested our method with an example involving fast angular motion (see Figure 9). Two cubes and spheres are placed into the spinning compartmentalized tray. The rotational motion makes each body move to a corner of the tray, to finally share a common spatial velocity due to the centrifugal force. With the high angular velocity of the tray, Baumgarte stabilization must work hard to stabilize constraints as the linear motion of the objects in the tray leads to interpenetration after each time step. As a result, while the objects largely appear to move without relative motion, the spatial velocities that come from the Baumgarte stabilized solve can become larger with faster rotations. To allow merging to occur in this

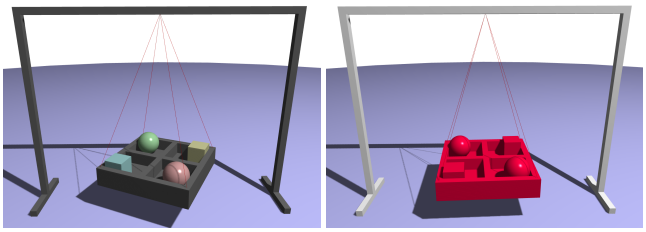

Fig. 9. Objects merge to a spinning compartmentalized box, though we observe that Baumgarte stabilization leads to velocities that do not satisfy the relative motion metric. Either smaller time steps, a merge threshold adjustment, or post step stabilization can be needed to produce merging in the presence of high angular velocities.

example, we can use smaller time steps (10 times smaller) or change the threshold. Likewise, we note that this issue can also be resolved with post-step stabilization (e.g., [Cline and Pai 2003]) provided that we compute the relative velocity before the stabilization step.

Contact ordering. In Figure 10, we emphasize the importance of contact ordering in the single iteration PGS solve. In this example, an impulse is applied to a body near the top of a tower. The impulse provides a force over only a single time step. If we do not order the contacts, the tower appears stiff in Figure 10 (right). When the contacts are ordered with respect to new interactions, i.e., the impulse, the result (middle) are close to ground truth (left). Indeed, ordering allows for the force information to be rapidly propagated through the blocks of the tower in the single iteration of the PGS algorithm. In some special cases, contact ordering can be insufficient. For instance, large mass ratios are a well known weak point for PGS. If the tower in Figure 10 has a large mass ratio (e.g., 100) between blocks of alternating layers, a single sweep with our contact ordering does not unmerge enough bodies from the collection for a correct result. Similarly, an upward impulse at the base of a tall tower can experience the same problem. While multiple sweeps can address problems with large mass ratios and large dense contact graphs, it would also be useful to explore the use of other techniques to deal with shock propagation in tricky cases [Erleben 2007].

8.2 Future Work

We observe that there are some straightforward improvements that are evident from the performance results, and useful extensions to consider in future work. We discuss some examples below. Note that Java implementations of our method, in both 2D and 3D, are hosted on GitHub.

Joint Constraints. Almost all rigid body simulators include joints constraints (e.g., rotary, spherical, etc.), but this is not something we included in our implementation, except for the electromagnetic body that turns contacts into bilateral rigid connection joints. Joint constraints are generally easier to handle than contacts because they are bilateral. To prevent numerical drift and eventual separation they require stabilization (which we already include for contacts). Including joints is a trivial extension that would require minimal changes to how we handle collections and constraints. In the case of joints, they would become internal to a collection on a merge,

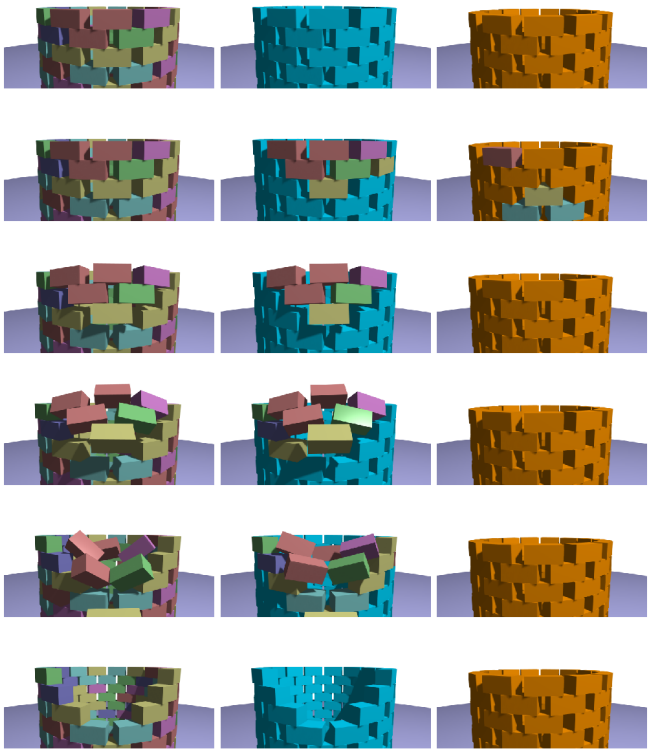

Fig. 10. An impulse is applied to a body at the top of the tower. Side by side comparison. (left) Ground truth, with no merging. (middle) Merging and contact ordering for the single iteration PGS solve. (right) Merging and no ordering of the contacts. Without contact ordering, nothing unmerges and the tower appears stiff. From top to bottom, $t = 0$, $t = 0.01$(impulse applied), $t = 0.5$, $t = 1.6$, $t = 4.7$, and $t = 14.1$. Collections are displayed in bright colors.

just like our contacts, and their constraint forces would be updated during the single iteration PGS step.

Collision Detection. Our implementation of BVH collision detection acceleration for collections uses incremental updates with naive tree rebalancing. This provides an important speedup in comparison to the minimal culling approach (i.e., the use of a single bounding volume for each collection), and is in particular valuable for the funnel example shown in Figure 8 (top). However, we note that the collision detection cost could still be improved if we could more efficiently cull bodies buried deep in merged collections.

Adaptive PGS sweep. We believe that computing less than one full PGS iteration per frame would be interesting for further lowering the computational cost in very large scale simulations such as the funnel in Figure 8. For instance, graph colouring could ensure that only a well distributed set of $1/N$ contacts are updated on every step, and the simultaneous breadth-first sweeps produced by impacts could be truncated at bodies where the influence (i.e., the velocity update) falls below a threshold. In contrast, the benefits of larger sweep counts are hard-to-identify, as our results are already close to ground truth (see Figure 10 and 12). In our experiments we tend

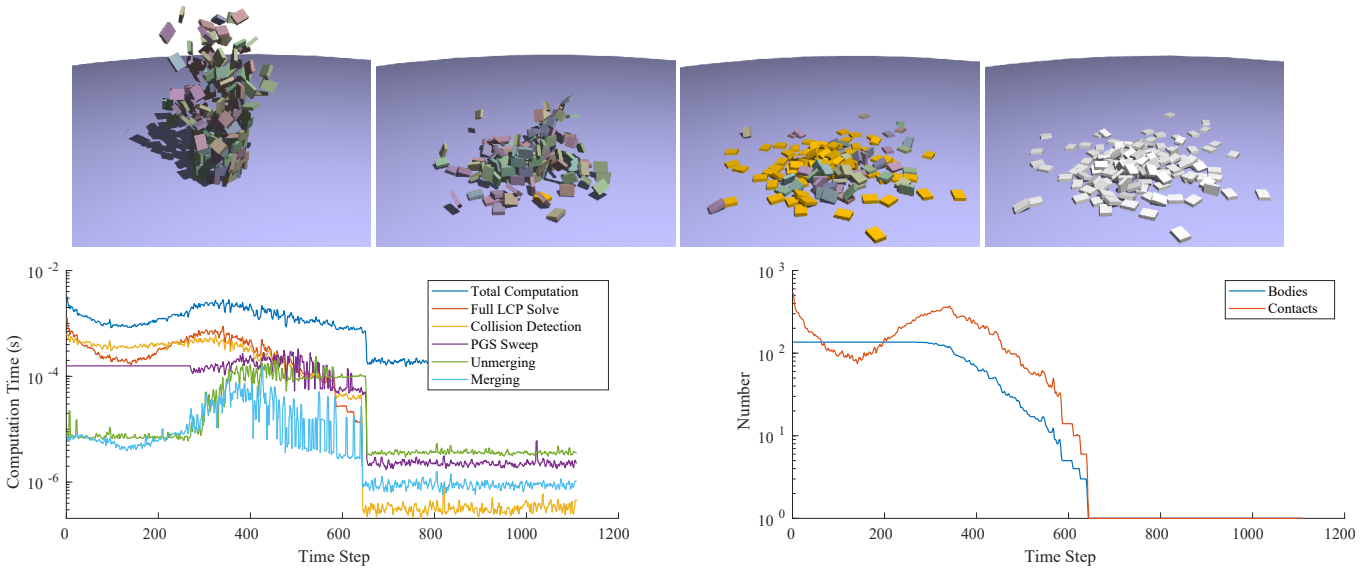

Fig. 11. Simulation of bodies falling onto the ground to form an unstructured stack (top row). Corresponding plots of compute time breakdown, and contact and body counts shown below. The bodies at rest first quickly merge with one another (orange) and with the ground, before finally falling asleep (light grey).

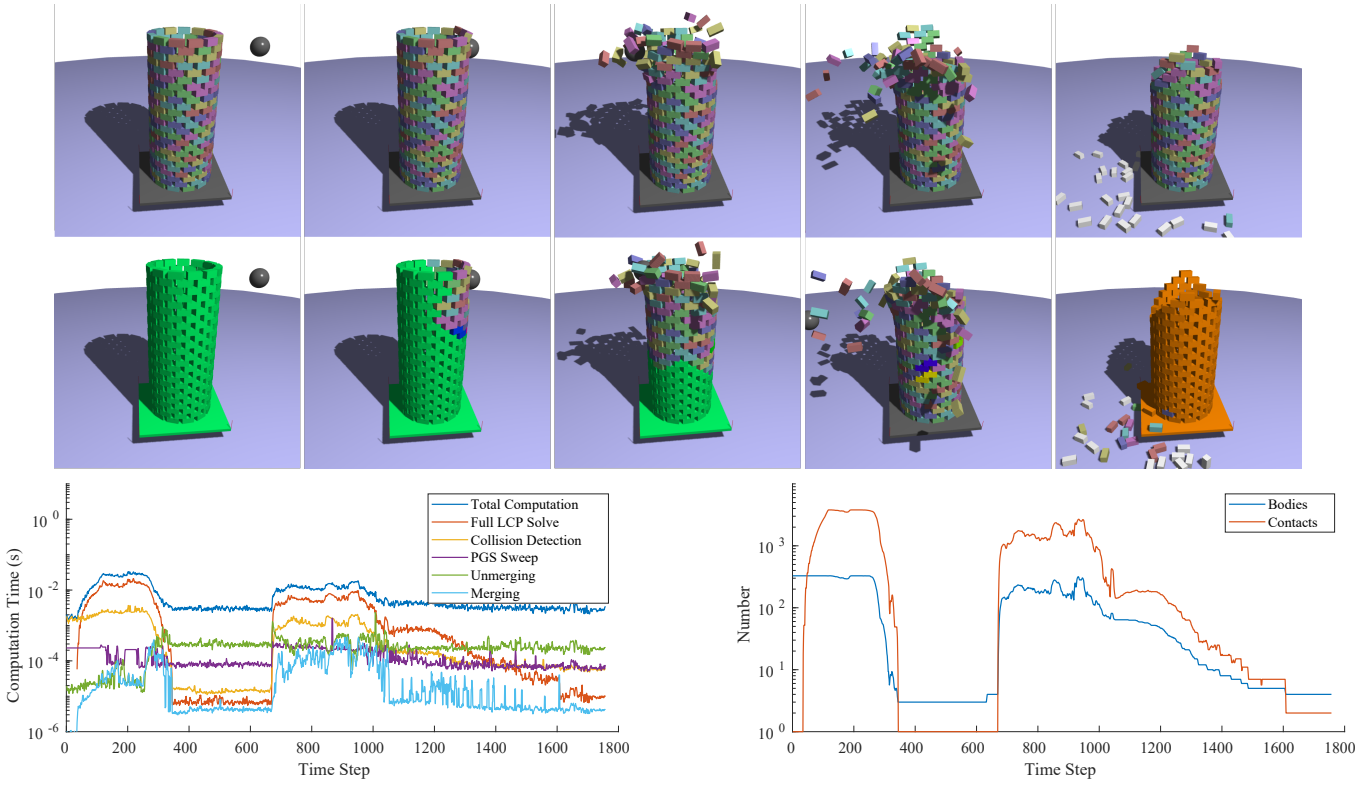

Fig. 12. Simulation of a tower standing on a floating platform hit by a projectile. (top row) Ground truth. (middle row) Results with our method. (bottom row) Corresponding plots of compute time breakdown, and the contact and body counts. Here, for accuracy, the full LCP is solved with 30 iterations instead of the 10 used in other examples. Collections are in bright colors and bodies that have fallen asleep on the ground plane are in light grey.

to observe conservative unmerging because the errors in the one iteration PGS sweep trigger unmerging (i.e., due to relative motion). In most of our examples, increasing iterations can primarily serve as a means of preventing unnecessary unmering. However, in special cases such as large-mass ratios and large dense contact graphs, one iteration of PGS may not be enough to identify all the bodies we should unmerge, as previously discussed in Section 8.1 *Contact ordering*.

Cycles and Complete Subgraphs. In 2D, if there is only one contact point between two bodies, it is likely that the configuration is unstable. When more bodies are involved, stability can arise when three bodies form a cycle with one contact between each. We experimented with identifying cycles of this form, where other bodies can participate in the cycle provided they have two contacts with one of their neighbors. In 3D, it is a bit more complicated, but we can instead identify stability with a complete subgraph of four bodies, with one contact between each pair of bodies. In future work, this observation can be used as an additional condition for merging bodies together. But this condition should also be handled carefully during the unmerging step. It will require keeping track of these cycles, because if we break a weld between two bodies that are part of a cycle (or a complete subgraph), the cycle of bodies may no longer be in a stable configuration. Thus, the other welds should be removed as well. While our 2D implementation explores this criteria, we we did not investigate this in 3D because we found that it leads to unnecessarily conservative policies for merging.

9 CONCLUSION

Multi-body systems with contacts are challenging to solve when large numbers of bodies are involved. The most computationally expensive steps are the resolution of contact forces and the detection of contacts. In this paper we have presented an algorithm that reduces computation time in both steps by merging collections of bodies when they share a common spatial velocity. This reduces the number of contacts being detected and solved in the dynamic system. Features of our contribution include, novel criteria for merging and unmerging based on relative velocity and monitoring of contact state, and the use of single iteration PGS to refine forces internal to collections, with a contact ordering for responsive unmerging. A sleeping strategy has been proposed on top of the merging that provides an additional speed-up when bodies are at rest by not performing the single iteration PGS in sleeping collections. We have demonstrated the efficiency of our method on several challenging simulation scenarios, which in each case produced plausible results. We believe that our general approach is not limited to multi-body systems, but with some adjustments could possibly be of benefit to position based dynamics solvers for the simulation of soft-bodies and fluids.

ACKNOWLEDGMENTS

We thank Marek Teichmann at CM Labs Simulation for suggesting the problem and feedback. We also thank Kenny Erleben and the anonymous reviewers for suggesting improvements to the paper. We gratefully acknowledge the support of CM Labs Simulations, and the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

R. Ando, N. Thürey, and C. Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article Article 103 (2013), 10 pages.

S. Artemova and S. Redon. 2012. Adaptively Restrained Particle Simulations. *Phys. Rev. Lett.* 109 (2012), 190201. Issue 19.

J. Baumgarte. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering* 1, 1 (1972), 1–16.

J. Bender, K. Erleben, and J. J. Trinkle. 2013. Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum* 33 (2013).

M. B. Cline and D. K. Pai. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *2003 IEEE International Conference on Robotics and Automation*, Vol. 3. 3744–3751.

C. Ericson. 2004. *Real-time collision detection.* CRC Press.

K. Erleben. 2004. *Stable, robust, and versatile multibody dynamics animation.* Ph.D. Dissertation. University of Copenhagen.

K. Erleben. 2007. Velocity-Based Shock Propagation for Multibody Dynamics Animation. *ACM Trans. Graph.* 26, 2 (2007), 12–es.

K. Erleben. 2018. Methodology for Assessing Mesh-Based Contact Point Methods. *ACM Trans. Graph.* 37, 3, Article 39 (2018), 30 pages.

M. Fratarcangeli and F. Pellacini. 2015. Scalable Partitioning for Parallel Position Based Dynamics. *Comput. Graph. Forum* 34, 2 (2015), 405–413.

E. Guendelman, R. Bridson, and R. Fedkiw. 2003. Nonconvex Rigid Bodies with Stacking. *ACM Trans. Graph.* 22, 3 (2003), 871–878.

S.-W. Hsu and J. Keyser. 2010. Piles of Objects. *ACM Trans. Graph.* 29, 6, Article 155 (2010), 6 pages.

T.-Y. Kim, N. Chentanez, and M. Müller-Fischer. 2012. Long Range Attachments - a Method to Simulate Inextensible Clothing in Computer Games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 305–310.

C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha. 2009. Fast BVH construction on GPUs. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 375–384.

S. Lee and A. Goswami. 2007. Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 4667–4672.

E. Lengyel. 2011. A Jitter-Tolerant Rigid Body Sleep Condition. In *Game Engine Gems 2*. AK Peters/CRC Press, 395–398.

P.-L. Manteaux, F. Faure, S. Redon, and M.-P. Cani. 2013. Exploring the Use of Adaptively Restrained Particles for Graphics Simulations. In *VRIPHYS 2013 - 10th Workshop on Virtual Reality Interaction and Physical Simulation*. 17–24.

P.-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M.-P. Cani. 2017. Adaptive Physically Based Models in Computer Graphics. *Comput. Graph. Forum* 36, 6 (2017), 312–337.

I. Millington. 2007. *Game physics engine development.* CRC Press.

B. Mirtich. 2000. Timewarp Rigid Body Simulation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 193–200.

M. Müller, N. Chentanez, M. Macklin, and S. Jeschke. 2017. Long Range Constraints for Rigid Body Simulations. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17)*. Article 14, 10 pages.

R. Narain, A. Samii, and J. F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6, Article 152 (2012), 10 pages.

E. G. Parker and J. F. O'Brien. 2009. Real-Time Deformation and Fracture in a Game Environment. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '09)*. 165–175.

A. Peiret, S. Andrews, J. Kövecses, P. G. Kry, and M. Teichmann. 2019. Schur Complement-based Substructuring of Stiff Multibody Systems with Contact. *ACM Trans. Graph.* 38, 5, Article 150 (2019), 17 pages.

S. Quinlan. 1994. Efficient distance computation between non-convex objects. In *IEEE International Conference on Robotics and Automation*. 3324–3329 vol.4.

S. Redon, N. Galoppo, and M. C. Lin. 2005. Adaptive Dynamics of Articulated Bodies. *ACM Trans. Graph.* 24, 3 (2005), 936–945.

H. Schmidl. 2002. *Opimization-based Animation.* Ph.D. Dissertation. University of Miami.

H. Schmidl and V. J. Milenkovic. 2004. A fast impulsive contact suite for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics* 10, 2 (2004), 189–197.

S. Stolpner, P. Kry, and K. Siddiqi. 2012. Medial Spheres for Shape Approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 6 (2012), 1234–1240.

R. Tonge, F. Benevolenski, and A. Voroshilov. 2012. Mass Splitting for Jitter-Free Parallel Rigid Body Simulation. *ACM Trans. Graph.* 31, 4, Article 105 (2012), 8 pages.

M. Tournier, M. Nesme, F. Faure, and B. Gilles. 2014. Seamless adaptivity of elastic models. In *Proceedings of Graphics Interface 2014 (GI 2014)*. 17–24.