

Statistical Simulation of Rigid Bodies

Shu-Wei Hsu and John Keyser

Department of Computer Science & Engineering
Texas A&M University

Abstract

We describe a method for replacing certain stages of rigid body simulation with a statistically-based approximation. We begin by collecting statistical data regarding changes in linear and angular momentum for collisions of a given object. From this data we extract a statistical "signature" for the object, giving a compact representation of the object's response to collision events. During object simulation, both the collision detection and the collision response calculations are replaced by simpler calculations based on the statistical signature.

Using this approach, we are able to achieve significant improvement in the performance of rigid body simulation. The statistical behavior of the simulation is maintained, including achieving valid resting positions. We present results from a variety of simulations that demonstrate the method and its performance improvement. The method is appropriate for rigid body simulation situations requiring significant performance improvement, and allowing for some loss in fidelity.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation; Simulation and Modeling [I.6.8]: Types of Simulation—Animation

1. Introduction

In the quest for more dynamic and interactive virtual environments, simulation of rigid bodies has played a critical role. Rigid body simulation has now become nearly ubiquitous in real-time graphics applications, and is an important component of many offline simulations. However, despite significant improvements in both simulation methods and in hardware performance, rigid body simulation can still take up a significant portion of the computational budget, particularly when there are large numbers of rigid bodies involved.

In many cases, exact fidelity within the simulation is not required. Sometimes, people have only a general sense of the way a simulation "should" look. For example, people expect a rolled die to bounce and tumble in a general direction, but have little sense of the exact response at every bounce. In other cases, a simulation has less visual impact, and an approximate result is sufficient. This may happen when a simulation is hidden from view, sufficiently far away from the viewer, or of sufficient complexity that a viewer cannot easily discern individual effects.

This insight has been the basis for a number of approaches to simplified simulation, and to concepts such as simulation levels of detail. However, most of these approaches have tended to simply take the same basic simulation framework and compute it with less fidelity (e.g. using a lower level of geometric detail). We propose a more fundamental change: rather than performing a full simulation, we instead replace key aspects of the simulation with results generated to mimic the statistical behavior of the simulation.

1.1. Overview

The most computationally significant portions of rigid body simulation are in the detection and response to collisions. In a typical system [BW01, GBF03], this process involves three stages: collision detection, determination, and response. Collision detection involves finding whether or not a collision has occurred for a rigid object. Collision determination, which is sometimes combined with collision detection or response, involves finding the exact point at which a collision occurs. Collision response involves computing the response of the object to the collision. A typical method for doing this is to compute an impulse that is applied to the object,

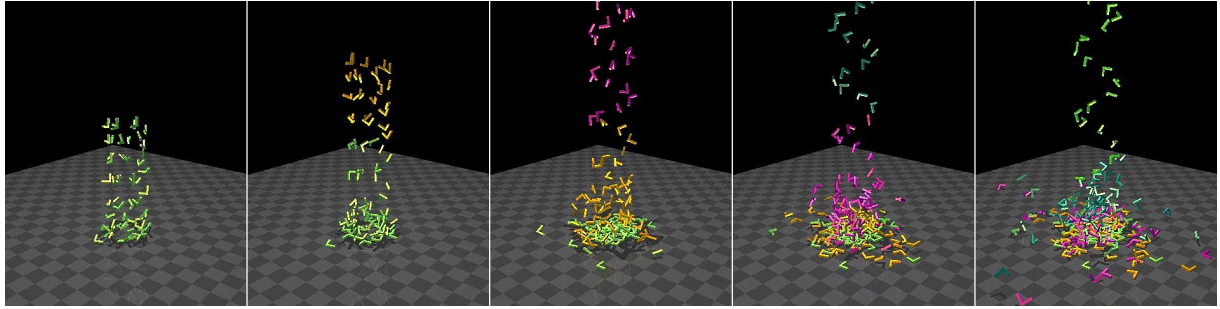


Figure 1: Frames from a statistical simulation of 300 L-Shape objects falling on the ground. Standard collision detection and response is replaced by a simplified statistically-based collision detection and response model.

changing its linear and angular momentum. Usually, collision detection and determination occupy the largest fraction of time in the simulation. Collision response is a relatively fast calculation, but relies on accurate collision determination in order to produce realistic results.

Our approach modifies each of these steps by using simplified calculations based on a statistical “signature” pre-computed for the object. Collision detection and determination are replaced by a simple collision detection computation based on a sphere, where the radius varies based on statistical analysis of typical collisions for the object. Collision determination is eliminated. While this alone provides most of our performance improvement, it poses challenges for collision response, since we no longer have an accurate point of collision. For collision response, we use statistical data to directly change linear and angular momentum, producing a plausible response. Furthermore, we directly modify the objects as they approach a rest state, in order to ensure that objects come to rest in configurations matching those commonly encountered statistically.

While this approach does lose some fidelity, the statistical properties of the rigid body motions are maintained. Further, these simplifications provide significant increases in simulation speed. This makes our approach highly suitable for any situation in which a tradeoff of fidelity for performance is desirable.

1.2. Main results

This work demonstrates that a statistically-based collision approach can provide plausible results and thus offer a useful tradeoff between simulation accuracy and speed. We fully describe how such a system can be implemented. In particular, we discuss below:

- How a statistical signature can be precomputed by analyzing several representative applications,
- How the statistical signature can be used to simplify the collision detection/determination/response calculations,

- How the particular case of rigid objects coming to rest (on a planar surface) is handled.

These key results are described in Section 3.

2. Prior Work

2.1. Simplified Simulation

Barzel et al. produced one of the earliest papers arguing for the need for plausible motion over precise motion [BHW96]. While their focus was somewhat different, in part arguing that less precision sometimes leads to *greater* plausibility, this basic theme was echoed by several later papers.

Chenney and Forsyth worked on ways to simplify simulations, including early work on certain types of simulations being replaced by statistical models [CF97, CF00]. Berka [Ber97] and Carlson and Hodgins [CH97] introduced the idea of simulation levels of detail (SLODs) to graphics-based simulation. Such approaches are designed to trade off simulation accuracy and speed, allowing simulations to vary in accuracy as needed. Our work would easily fit into such an SLOD framework.

There has also been a push toward simplifying simulation by modeling modes of response. James developed some of the key techniques related to modeling of deformations using dimensionality reduction [JF03, BJ05]. This work was later followed up by Treuille et al. [TLP06], who applied it to fluid simulation. This sense of reducing computation in real-time by precomputing animation and extracting a simplified model of behavior has motivated the approach we explore here.

The work presented here does not attempt to investigate the perceptual effects of simplified simulation, determining how significant the loss of fidelity is. This is, as a whole, an underexplored area, although O’Sullivan and others have done a significant amount of work in coupling such perceptual studies with simulation [OD01, ODGK03].

2.2. Simplified Collisions

Collision detection and response is a well-explored field. Rather than reviewing it extensively here, we suggest the reader refer to one of the many research papers (e.g. [MC95]), survey papers (e.g. [TT01]), courses (e.g. [BW01]), or books (e.g. [AMHH08]) that discuss collision detection and impulse-based response.

Simplified collision detection is a very common approach used in the real-time graphics community. Typically, in order to achieve the frame rates desired, object collisions are computed with far simpler geometry (e.g. a sphere, either bounding or not). A bounding volume hierarchy may be used, but evaluated only to a particular depth, in order to maintain a good frame rate [Hub96]. However, in the cases where such approaches are used, the collision detection is usually all that is desired (e.g. to find if a character has hit some other object). For rigid body simulation, more is needed - the point(s) of collision and from that the response must be calculated. To achieve plausible collision response, the simplified collision detection approach is not sufficient.

Although the principles may have wider applicability, our work focuses on rigid body simulation. For deformable objects, other approaches for speeding up simulations (e.g. via conservative bounding spheres) have been explored [JP04].

Work on simplified collision response is far more limited. One approach has been to use particle systems instead of rigid bodies [AECC05, Jak03], but here one must maintain several particle-particle linkages, which can be problematic. Collisions have also been approximated for control, rather than for simulation speed. Popović et al. [PSE*00] modified collision responses in order to control simulation behavior, and James and Twigg [TJ08] used modified collisions to “reverse” simulations from a solved steady state.

Like earlier approaches, our method uses a simplified sphere model (in our case a single sphere) for collision detection. Unlike prior methods, we allow our sphere to vary in size based on the object state and precomputed information about the object. We avoid the necessity of accurate collision determination by replacing impulse-style collision response with a statistically based response model.

3. Methodology

3.1. Overview

Our approach has two stages. In the first stage, which is pre-computed, we run several simulations for individual objects from a variety of initial conditions. We use these results to collect a statistical “signature” for that object’s response to collisions. In the second stage, we use the statistical signature during run-time to quickly compute collisions and responses of that object type. When objects are approaching a rest condition, we modify our approach slightly in order to ensure the object ends in a valid rest state.

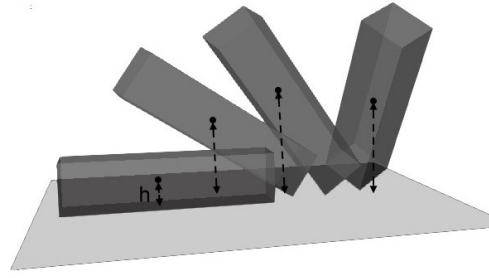


Figure 2: Illustrating the collision height. Height measures the distance of the center of mass from the plane an object is colliding against.

3.2. Determining a Signature

We consider an object’s *signature* to be a collection of statistical data regarding its behavior under collision conditions. We determine a signature by running numerous tests from various starting configurations, measuring the values we are looking for directly, and then extracting both the mean, μ , and standard deviation, σ , for those statistics. There are three main parts to the signature, which we discuss here.

3.2.1. Determining Sphere Radius

Perhaps the most important part of the signature is the radius of the sphere to be used for collision. During our simplified simulation, we will replace the object geometry with a simplified sphere geometry. We must determine the appropriate radius of sphere to use. To do this, we determine height, the distance of an object’s center of mass from a plane at the time of collision. Figure 2 illustrates the height measurement for a box.

Height can vary significantly, depending on the dimensions of the object. Furthermore, height is correlated very closely with angular velocity, ω . Considering the box example from Figure 2, the minimum height shown can only occur when the box collides exactly along its large face. Generally this requires that the box be falling straight down with no angular velocity. As ω increases, it is not possible for the collision to occur at this height, since the object will rotate as it approaches the plane. One of the corners or edges of the box will collide before the face can collide.

This is illustrated in Figure 3 for a 2D square (side length 1). The graph shows the range of recorded collision heights recorded from a series of experiments with random starting heights at a variety of angular velocities. With no angular rotation, the height at collision may be anywhere from 0.5 to $\sqrt{2}/2$. As angular rotation increases, the height approaches $\sqrt{2}/2$. For a given angular velocity, then, we can compute the mean (μ) and standard deviation (σ) of the heights over a variety of simulations. We store this in a table. While this simple case could have been computed analytically, it illustrates the process used for more complex shapes. Further,

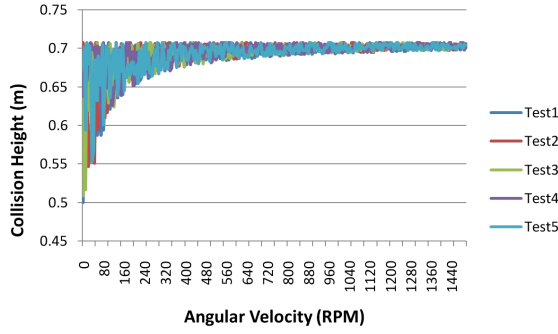


Figure 3: Graph showing the range of collision heights for a 1×1 square falling against a line. Angular velocity is measured relative to a fixed linear velocity. As the angular velocity increases, the collision height approaches $\sqrt{2}/2$.

while this may seem obvious for the simple 2D square, similar behavior is seen for more complex objects in 3D.

We call the height at collision h_{coll} . To collect h_{coll} data, we run several trials for a given object. We fix the linear velocity, v , and use a variety of random starting orientations and angular velocities (ω). Note that ω must be recorded relative to the magnitude of v (i.e. it is not the absolute rotation, but rather the rotation relative to linear velocity that matters). As a result, this table is indexed by $\|\omega\|/\|v\|$. Given a range of ω values, extracting μ and σ for that range is straightforward. We store h_{coll} in a table containing the μ and σ values indexed by relative velocity (in the end, only the μ value is used, so σ may be eliminated).

As can be seen in Figure 3, the height tends to vary much more for lower values of ω than for larger ones. Thus, to assure that we capture the variation in μ and σ accurately, we use a non-uniform sampling over the range of ω values. Specifically, we sample ω more closely at lower values, and much more sparsely at higher values. Later, we will assume that we can reconstruct μ and σ for any ω value by interpolating between the nearest samples.

3.2.2. Determining Collision Response

The second key part of the signature is the characterization of the collision response. Collision response characterizes the result of a collision in terms of the change in v and ω due to the collision. Again, we will perform a series of trials, and determine μ and σ for various indexed values in the table. Each trial begins with a random starting orientation at a fixed height from a plane, and initial v and ω are set (v is set so the object moves toward a plane; no gravity or external forces are applied).

We collect data in a table indexed by four different values. These are illustrated in Figure 4a, where the values of v and

ω before collision are represented as V^- and Ω^- . Specifically, we sample for different values of:

- θ , the angle between V^- and the inverse direction of the plane normal.
- ϕ , the angle between V^- and Ω^- .
- $\|V^-\|$, the magnitude of linear velocity.
- $\|\Omega^-\|$, the magnitude of angular velocity.

Again, we do not have to sample these values uniformly, but can adjust the sampling to better capture wider variations in the measured statistics.

We use an impulse-based simulation with precise collision detection to determine the collision location and response. From this, we compute three response values:

- V_{Rigid}^+ , the linear velocity after collision.
- Ω_{Rigid}^+ , the angular velocity after collision
- $V_{Particle}^+$, the resulting linear velocity we would expect if the object were simply a sphere (or particle) rather than a rigid object.

We collect and store data in two tables (each indexed in the four dimensions listed earlier). One table (the Ω table) contains μ and σ values for the angular velocity (Ω_{Rigid}^+). The other table (the V table) contains μ and σ values for the *difference* in linear velocity from a particle response, i.e. $(V_{Rigid}^+ - V_{Particle}^+)$. While typically the μ value for that table is near 0 (i.e. on average, the response is similar to a particle bouncing), the σ value certainly is not. Note that this is one of the key areas where our statistical signature provides additional information that could not be obtained by simply treating the whole object as a sphere.

3.2.3. Determining Resting Configurations

The third aspect of the signature that we collect is information about the typical resting conditions (on a plane) for an object. As an object comes to rest, there are typically only a few stable configurations it can end up in. Mirtich previously explored the idea of estimating common resting configurations (or in his terms, “pose statistics”) [Mir96], however most previous work has focused on finding such stable configurations automatically from the geometry (and this is not always an easy operation). For some models, even though some configurations might technically be stable, those configurations rarely occur in practice (e.g. if small forces would tend to upset the stable configuration). An example would be a long skinny box: though it could end up resting on one end, it is far more likely to end on one of its larger sides. Thus, we again measure resting conditions using a statistical approach.

We perform several simulations, in a variety of starting configurations (differing orientations and velocities), but apply gravity to the simulation. Again using an impulse-based simulator, we simulate the object precisely until it reaches a resting configuration (momentum falls below a threshold). We collect two pieces of data:

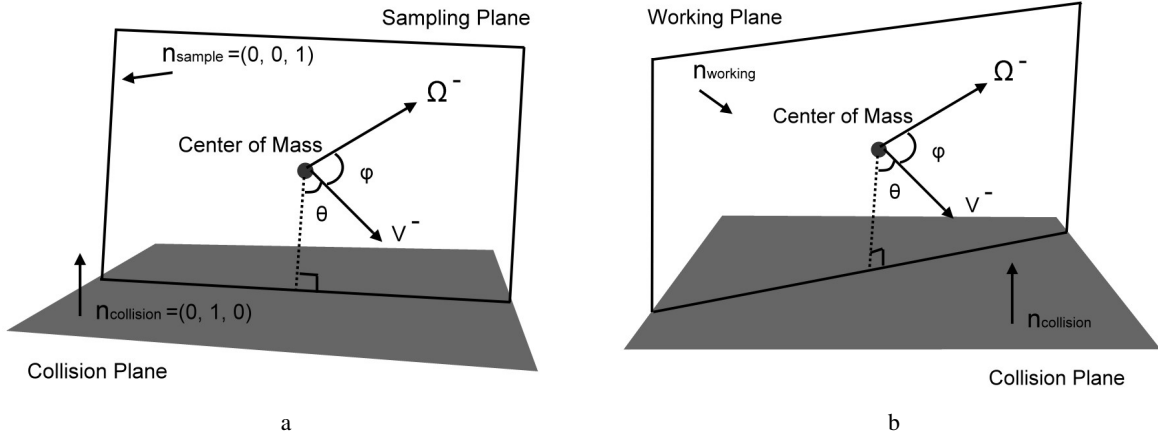


Figure 4: Illustration of the sampling plane (a) on which data is collected. V^- is the linear velocity before collision, by definition in the sampling plane, moving at an angle θ to the plane normal. Ω^- is the vector expressing angular velocity (the direction giving axis of rotation, and the magnitude the absolute velocity). It is not usually in the sampling plane, and the vector forms an angle ϕ with V^- . When the data is used, the appropriate forces are computed and applied relative to the working plane (b).

- The height, h_{rest} , of the center of mass when the object is at rest. Note that this will often be much smaller than the μ of h_{coll} at $\omega = 0$.
- The rest orientation n_{rest} , giving the plane normal expressed in the object's local coordinate system.

Notice that each n_{rest} should correspond to a unique h_{rest} . Also, n_{rest} represents the “up” direction for the object in that rest configuration; rotating around that vector should still give a stable rest configuration.

In this case, we do not compute μ and σ values. Rather, we run several trials, and determine the $(n_{\text{rest}}, h_{\text{rest}})$ values. For those that occur often enough (e.g. more than 5% of the time), we store this as a “valid” rest configuration within the object signature.

3.3. Using Signature Data to Approximate Collisions

Given the signature data for a model, we can use it at run-time to simplify our collision calculations. We will describe our basic approach here, and discuss how we handle resting conditions in section 3.4.

3.3.1. Determining Object Collision

At run-time, we replace the geometry of the object with a simple sphere, and use this to test for collisions. The same sphere is used for both object-plane and object-object collisions. To determine the radius of the sphere to use (spheres are always centered on the center of mass), we compute the relative magnitude of the angular velocity to the linear velocity, i.e. $\|\omega\|/\|v\|$. We use this to index into the table (both

interpolating neighboring values and simply using the nearest value work well) of h_{coll} . This table gives us the mean value, μ , of the collision distance, and we use that for the radius of the sphere. Note that the sphere radius may change from timestep to timestep; since this change is usually minor, we can limit the frequency of sphere size updates if desired (we have not done this in our implementation results presented below).

When an object is found to collide with another object or with a plane (polygon), we apply the collision response calculation described next.

Note that this approach is focused on the narrow phase of collision detection. It can be easily augmented by standard broad-phase collision detection speedups, such as a sweep-and-prune operation or spatial subdivision. It is best to use a sphere of the maximum size possibly needed (i.e. a bounding sphere) for the broad-phase calculations, since otherwise temporal coherency may be lost due to updates of sphere sizes.

3.3.2. Determining Object Response

To calculate collision response, we make use of the V and Ω tables stored in the signature to determine a statistically plausible collision response. Refer to figure 4b for some of the notation. The specific steps are as follows:

1. Calculate the collision plane normal. If the object collides with a plane, that plane is the collision plane. If the collision is with another object (represented by a sphere), the plane is the separating plane between the two spheres

at the point of collision (perpendicular to the line connecting the sphere centers). Note that only the collision plane normal, $n_{collision}$, is needed in our scheme (we do not need the actual plane or the penetration depth of the object pair).

2. *Calculate the working plane normal.* We define the working plane to be the plane spanned by the collision plane normal and the object's linear velocity; its normal is $n_{working}$.
3. *Compute a rotation matrix from the canonical system to the working system.* The data stored in the V and Ω tables is computed relative to the canonical system used for sampling (see Figure 4a). We must be able to convert that data to the system (defined by the collision plane and working plane normals) encountered at runtime (see Figure 4b). This rotation matrix (formed from the collision and working plane normals) is applied to the results of the later steps (we do not explicitly state this later).
4. *Compute the current configuration.* For the working system, we need to compute the values of θ , ϕ , $\|V^-\|$, and $\|W^-\|$. These are computed as in the signature determination stage for collision with a plane. For collision with another object, the velocity is determined relative to the other object (i.e. as if that object were fixed).
5. *Retrieve statistical data.* The configuration data is used to retrieve values (μ_V, σ_V) and (μ_W, σ_W) from the V and Ω tables.
6. *Determine the new Ω^+ .* We form a Gaussian distribution based on μ_W and σ_W , and we then generate a random sample from that distribution.
7. *Determine the new V^+ .* We first calculate a linear velocity response, $V_{particle}^+$ based on a simple calculation as if the object were a particle. Then, similar to Ω , we generate a random value, $V_{relative}^+$ based on μ_V and σ_V . The final value is then determined as $V^+ = V_{particle}^+ + V_{relative}^+$.
8. *Update v and ω .* The linear and angular velocity are simply replaced by the new V^+ and Ω^+ .

For object-object collision response, it would be impractical to sample and store data from all possible collision configuration pairs. Instead, we approximate these collisions using the object-plane response data. For each collision pair, we first compute the collision plane (working plane) between the spheres approximating each object. We then apply the object-plane collision response for the two objects as if they were colliding with that plane. The two object responses are computed independently.

The end result is that the object responds to the collision by receiving a new linear and angular velocity that fall within the statistical range of observed responses for similar initial states.

3.4. Achieving Resting States

The above approach works well for most dynamic scenes, but has a serious deficiency when objects come to rest. In

particular, there are no restrictions on the final orientation of the object, so objects may come to rest in awkward and invalid states. Although people will often not notice this until it is pointed out, and it may make no difference depending on level of detail required for the simulation (e.g. it may be so far away this makes no difference), there will be cases where we wish to ensure that the resting states of the objects are valid. We do this by applying "resting conditions" that explicitly move the object to a valid rest states.

We ignore resting conditions until we detect that an object has sufficiently low linear and angular momentum, within a certain distance from a horizontal plane. For objects that come to rest in other situations (e.g. if the objects are piled up on top of others), we do not do anything special; once the object has come to a near-still state, we set its velocities to zero and consider it at rest. In such cases, objects do not have a limited well-defined set of rest poses, anyway, so the need for applying our resting conditions is much less.

For the objects that we wish to apply resting conditions to, we first determine the nearest n_{rest} value from the signature, given the current configuration (we choose the one requiring the smallest angular rotation from the current orientation). We then gradually modify the object so that it moves toward this rest state.

One part of the modification is replacement of the sphere radius from the table with the h_{rest} corresponding to the chosen n_{rest} . A sudden replacement may cause the object to suddenly "drop" down (since h_{rest} is usually smaller than h_{coll}), so we transition to this radius gradually over several frames of animation.

The other part of the modification is the motion toward the appropriate orientation. We apply a small rotational impulse to the object at each collision that will move it toward the n_{rest} orientation. This is done instead of the Ω^+ calculation described earlier. Since resting conditions are only applied when the object's motion is already small, this change is usually unnoticed. The effect is that the object will gradually bounce toward its stable resting state.

Together, over a series of collisions as the object is coming to rest, it will slowly turn to a valid resting condition. When it finally does come to rest (dropping below an even smaller threshold of linear and angular velocity), it should be in a valid configuration.

4. Implementation and Results

We have implemented the approach described above, and tested it on a variety of models. We will first discuss our implementation, and then present examples and timing results. Examples may be seen in both the figures and the accompanying video.

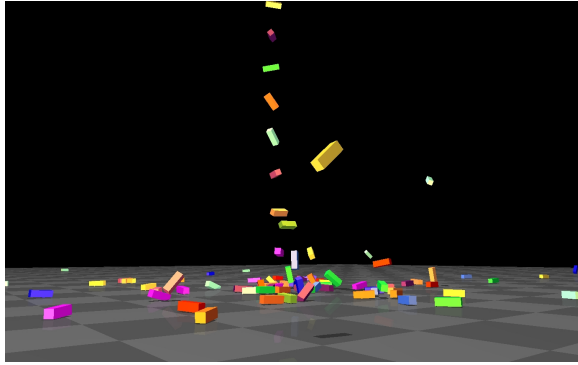


Figure 5: Statistical simulation of 100 boxes.

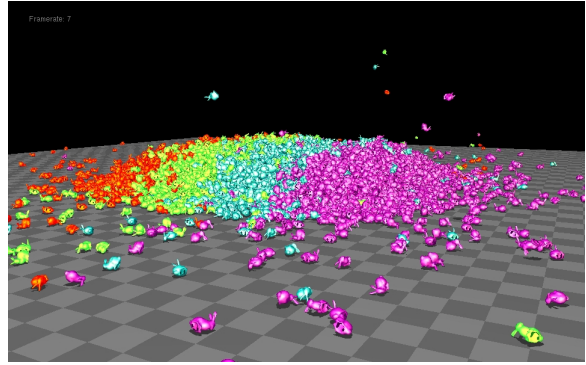


Figure 6: Statistical simulation of 10000 bunnies.

4.1. Implementation

We first developed a fairly standard impulse-based rigid body simulation system. This same simulation system is used as the basis for the comparisons and timing evaluations described below, as well as for collecting signature data. For collision detection, the SWIFT++ [EL01] library was used. Collision response was calculated through an impulse-based response. We will refer to this system below as the *impulse* system.

In our simplified simulation, which we will refer to this system as the *statistical* system, we followed the approach described above. For collecting statistical data, we ran approximately 30 trials for each configuration for which we extracted μ and σ . The total time spent collecting signature data was approximately 6-8 hours.

The h_{coll} information was collected at 16 sample values of ω . For the V and W tables, we sampled θ at 7 values, ϕ at 10 values, $\|V^-\|$ at 26 values, and $\|W^-\|$ at 16 values. This required a total of 873,600 simulations. As for resting conditions, we found 4 resting states for the box model we demonstrate, and 4 resting states for the bunny model. The total storage size for the signature information from a model was approximately 3 MB.

The SWIFT++ system uses a sweep-and-prune approach [CLMP95] in the broad phase to limit the number of collision checks between objects. While this approach makes little difference with small numbers of objects, it is important when dealing with large numbers of objects. We therefore implemented a bounding-sphere-based sweep-and-prune approach in our simulation, also, in order to provide a better comparison. Also, we did not implement any further broad-phase algorithms, such as spatial subdivision, since it did not appear that SWIFT++ provided this.

4.2. Results

We present some results of our work in Figures 1, 5, 6, and 8. More illustrative results can be seen in the accom-

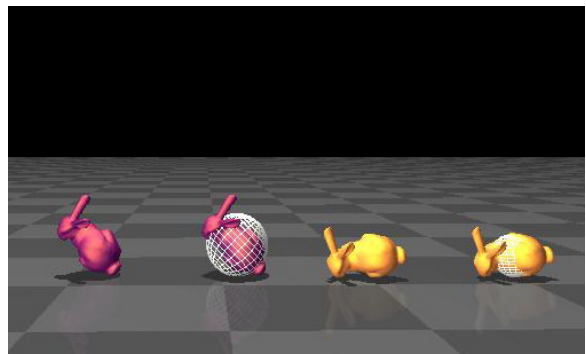


Figure 7: At left, the pink bunny model has come to rest without resting conditions applied, and at right the yellow bunny is shown with resting conditions applied. The same model is shown both with and without is colliding sphere. Note that the resting conditions result in the bunny having a more natural final orientation and collision sphere.

panying video. We show results from four basic shapes. The box examples are presented as a “base” case. We use both bunny and armadillo-man shapes to demonstrate the ability to handle more complex objects. Finally, we include examples of an ‘L’-shaped object, which we believe should be a “worst case” example for our system (due to the different dimensions and concavity). The overall behavior is such that in informal polls, most were unable to tell which one was “correct” for several simulations (L-shape simulations), and sometimes could only determine this when the types of errors to look for were pointed out.

Figure 1 shows the result of a statistical simulation of 300 L-Shape dropped from a cylinder pattern. Figure 6 shows a frame from a statistical simulation of 10000 bunnies. Figure 8 shows a comparison of the impulse and statistical simulations.

A close-up view of the effect of resting conditions is demonstrated in Figure 7. Without resting conditions, the

bunny model appears in an awkward orientation, and the sphere radius used for collision is too large (the bunny is actually slightly off the floor). Other models may look even worse. With resting conditions, the bunny model appears in a more natural orientation. The collision sphere has a radius appropriate to keep the model in contact with the floor.

4.3. Performance

In a variety of test situations, we achieve a clear improvement in speed with the simplified model versus all compared alternatives. To show the results, we provide a comparison of running times for a set of n randomly dropped bunnies in Table 1. Note that all methods implemented a full sweep-and-prune computation for the broad phase of collision detection.

For both the impulse and the statistical simulations, the vast majority of time was spent in the collision detection stage (this includes collision determination in the impulse system). The time spent on collision response, and in the statistical simulation on resting conditions, was negligible (typically orders of magnitude lower) than the time spent on collision detection.

As the table shows, compared to full impulse simulations of both the full model and a decimated model, our statistical simulation performs far faster. We also compare to a simplified model based on replacing the bunny model with a set of 7 spheres. While in the best case we would anticipate timings as much as 7 times faster (since we have only a single sphere check, instead of seven), the overhead of the broad phase of collision detection (shared by both approaches) reduces our benefit somewhat. However, for both small and large simulations, our method still performs noticeably faster than the 7 sphere models. Also, note that while the 7 sphere example usually produces reasonable collision responses, it does not necessarily achieve reasonable rest states, as in our method.

5. Conclusion

We have presented a method for replacing rigid body simulation by a statistically-based simulation. The results achieve significant performance improvement, primarily by simplifying collision detection. Objects are replaced by spheres with a radius based on statistical analysis of object collisions. Since the simplified collision detection cannot provide the necessary accuracy of contact points necessary to achieve plausible collision response, we use a statistically based simulation response model to create reasonable collision responses. To avoid awkward resting states for objects, we force them into orientations that were recognized as valid from an earlier set of simulations.

5.1. Comparison and Evaluation

Our results show that our method can provide major performance improvements. However, it is important to note that

our approach carries significant tradeoffs. In particular, because the responses of objects obey only statistical response properties, close examination of collisions can result in observable inconsistencies (such as brief interpenetrations of objects or unrealistic response). Furthermore, collision responses that rely on objects being in very particular configurations (e.g. axis aligned starting positions) may not be faithfully captured. However, in exchange for this lack of fidelity we are able to produce responses that obey all statistical properties, and can be computed at much higher frame rates. For situations where a simulation forms the main focus of an animation, the method may not be appropriate, since small errors will be rapidly noticed. But, in situations where the simulation is either sufficiently obscured/unimportant/less important, etc., our method should work well.

As the number of objects increases significantly, one would expect the broad phase of collision detection to dominate the narrow phase. While our statistical approach should still be faster than the alternatives, the speed benefit will be reduced somewhat, and thus the quality/speed ratio may shift in favor of alternative methods. Still, as we have shown in our timings, for examples that are feasible in real-time currently and in the near future, our approach should still provide a clear benefit over the alternatives.

5.2. Limitations

There are several important limitations of our method. Clearly, there is a significant loss of accuracy, in exchange for improvements in time. This is perhaps most evident in the handling of object-object collisions. Collecting data from object-object collision examples is much more challenging than for the object-plane case, due to the vast number of possible configurations between the two objects. Our simplification — using the object-plane response — will lead to artifacts that may be quite noticeable in some situations. One such artifact is that after a collision response, the two objects will always move away from each other. This can lead to objects seeming to “freeze” in space temporarily. Especially in large collections of objects, stacking may look unrealistic as a result of the object-object collision artifacts at the pairwise level.

For performance, it should be noted that our method has only limited (though still noticeable) performance improvement over a competing ultra-low resolution model, such as the 7-sphere bunny example. A key advantage of a model such as the 7-sphere one is that it does not involve the extensive precomputation of our method, and will have a much lower memory footprint. This lower memory footprint can be particularly important for certain applications. However, it should be noted that generating good ultra-low resolution models usually requires tedious manual specification and tweaking to get a good result.

Table 1: Performance improvement for the statistical simulation vs. various other simulations. For each simulation, one step refers to the total amount of time taken for a simulation step. Collision is the portion of the time spent in collision detection and determination, and response is the portion of the time spent in collision response. Times are in seconds, and the percentages are the amount of time relative to the impulse response (or the 7-sphere example for 10,000 objects). We compare four different approaches: a full impulse simulation of the original bunny model (with 2503 vertices and 4698 faces), a full impulse simulation on a decimated model (with 99 vertices and 194 faces), a full impulse based simulation of the bunny approximated by seven spheres, and a statistical simulation described here.

# of objects	type	impulse (sec.)		low (sec.)		7-sphere (sec.)		statistical (sec.)	
10	one step	0.00017861	100.00%	0.00004699	26.31%	0.00001974	11.05%	0.00000453	2.54%
	collision	0.00017535	100.00%	0.00004363	24.88%	0.00001561	8.90%	0.00000061	0.35%
	response	0.00000018	100.00%	0.00000035	201.43%	0.00000032	184.29%	0.00000021	120.00%
100	one step	0.00472734	100.00%	0.00098305	20.79%	0.00015952	3.37%	0.00005729	1.21%
	collision	0.00466885	100.00%	0.00093947	20.12%	0.00012011	2.57%	0.00002694	0.58%
	response	0.00003072	100.00%	0.00002231	72.62%	0.00001914	62.30%	0.00000566	18.44%
1000	one step	0.92583944	100.00%	0.08142146	8.79%	0.00924859	1.00%	0.00691050	0.75%
	collision	0.92304164	100.00%	0.08050417	8.72%	0.00712144	0.77%	0.00587160	0.64%
	response	0.00255375	100.00%	0.00067762	26.53%	0.00187502	73.42%	0.00077710	30.43%
10000	one step					0.33261309	100.0%	0.21312389	64.08%
	collision					0.28589991	100.0%	0.20217268	70.71%
	response					0.04345081	100.0%	0.00800835	18.43%

5.3. Further Work

There are a number of ways that this work can spur further work in the future. The method we have proposed here could certainly be augmented in various ways, including determining an “optimal” sampling of the domain during signature gathering, or better handling of highly non-convex shapes. Eventually, the approach would certainly benefit from a rigorous perceptually-based study, as well.

We believe that a statistical approach as described here has potential wider application, such as to broad-phase collision detection and to fluid simulation (other statistical approaches have already been applied to fluid simulation). There are also additional simplifications that might be made to massively large simulations, based on statistical packing measurements.

Finally, the work presented here discusses determination of the signature from an impulse-based simulation. However, it may be possible to collect signatures via other means, including measurement of real-world objects. Our work has demonstrated that a statistical model of collision response is sufficient to produce feasible results. This, then, opens up much wider simulation possibilities, including incorporation of real-world and not just virtual objects.

References

- [AECO05] ATENCIO Y., ESPERANCA C., CAVALCANTI P., OLIVEIRA A.: A collision detection and response scheme for simplified physically based animation. In *SIBGRAP, the 18th Brazilian Symposium on Computer Graphics and Image Processing* (2005), pp. 291–298. 3
- [AMHH08] AKENINE-MÖLLER T., HAINES E., HOFFMAN N.: *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008. 3
- [Ber97] BERKA R.: Reduction of computations in physics-based animation using level of detail. In *13th Spring Conference on Computer Graphics* (1997), pp. 69–76. 2
- [BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Proc. Eurographics Workshop Computer Animation and Simulation* (1996), Springer, pp. 183–197. 2
- [BJ05] BARBIĆ J., JAMES D. L.: Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.* 24, 3 (2005), 982–990. 2
- [BW01] BARAFF D., WITKIN A.: *Physically-Based Modeling*. Tech. rep., SIGGRAPH Course Notes: pixar.com/companyinfo/research/pbm2001/, 2001. 1, 3
- [CF97] CHENNEY S., FORSYTH D.: View-dependent culling of dynamic systems in virtual environments. In *Proceedings of Symposium on Interactive 3D Graphics* (1997), pp. 55–58. 2
- [CF00] CHENNEY S., FORSYTH D. A.: Sampling plausible solutions to multi-body constraint problems. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 219–228. 2
- [CH97] CARLSON D. A., HODGINS J. K.: Simulation levels of detail for real-time animation. In *Proceedings Of Graphics Interface* (1997), pp. 1–8. 2
- [CLMP95] COHEN J. D., LIN M. C., MANOCHA D., PONAMGI M.: I-collide: an interactive and exact collision detection system for large-scale environments. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), ACM, pp. 189–ff. 7
- [EL01] EHMANN S., LIN M.: Accurate and fast proximity queries between polyhedra using surface decomposition. *Computer Graphics Forum (Proceedings of Eurographics)* 20, 3 (2001), 500–510. 7

- [GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Non-convex rigid bodies with stacking. *Transactions on Graphics* 22 (2003), 871–878. Proceedings of SIGGRAPH 2003. 1
- [Hub96] HUBBARD P. M.: Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.* 15, 3 (1996), 179–210. 3
- [Jak03] JAKOBSEN T.: *Advanced Character Physics*. Tech. rep., Gamasutra.com Resource guide: gamasutra.com/resource_guide/20030121/jacobson_01.shtml, 2003. 3
- [JF03] JAMES D. L., FATAHALIAN K.: Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics* 22 (2003), 879–887. 2
- [JP04] JAMES D. L., PAI D. K.: Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.* 23, 3 (2004), 393–398. 3
- [MC95] MIRTICH B., CANNY J.: Impulse-based simulation of rigid bodies. In *SIGGRAPH '95: Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), ACM, pp. 181–ff. 3
- [Mir96] MIRTICH B.: *Impulse-Based Dynamic Simulation of Rigid Body Systems*. Ph.D. thesis, Computer Science, University of California at Berkeley, Berkeley, CA, 1996. (See Section 7.5). 4
- [OD01] O’SULLIVAN C., DINGLIANA J.: Collisions and perception. *ACM Trans. Graph.* 20, 3 (2001), 151–168. 2
- [ODGK03] O’SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. *ACM Trans. Graph.* 22, 3 (2003), 527–536. 2
- [PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A.: Interactive manipulation of rigid body simulations. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 209–217. 3
- [TJ08] TWIGG C. D., JAMES D. L.: Backward steps in rigid body simulation. *ACM Trans. Graph.* 27, 3 (2008), 1–10. 3
- [TLP06] TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. *ACM Trans. Graph.* 25, 3 (2006), 826–834. 2
- [TT01] THOMAS F., TORRAS C.: 3d collision detection: A survey. *Computers and Graphics* 25 (2001), 269–285. 3

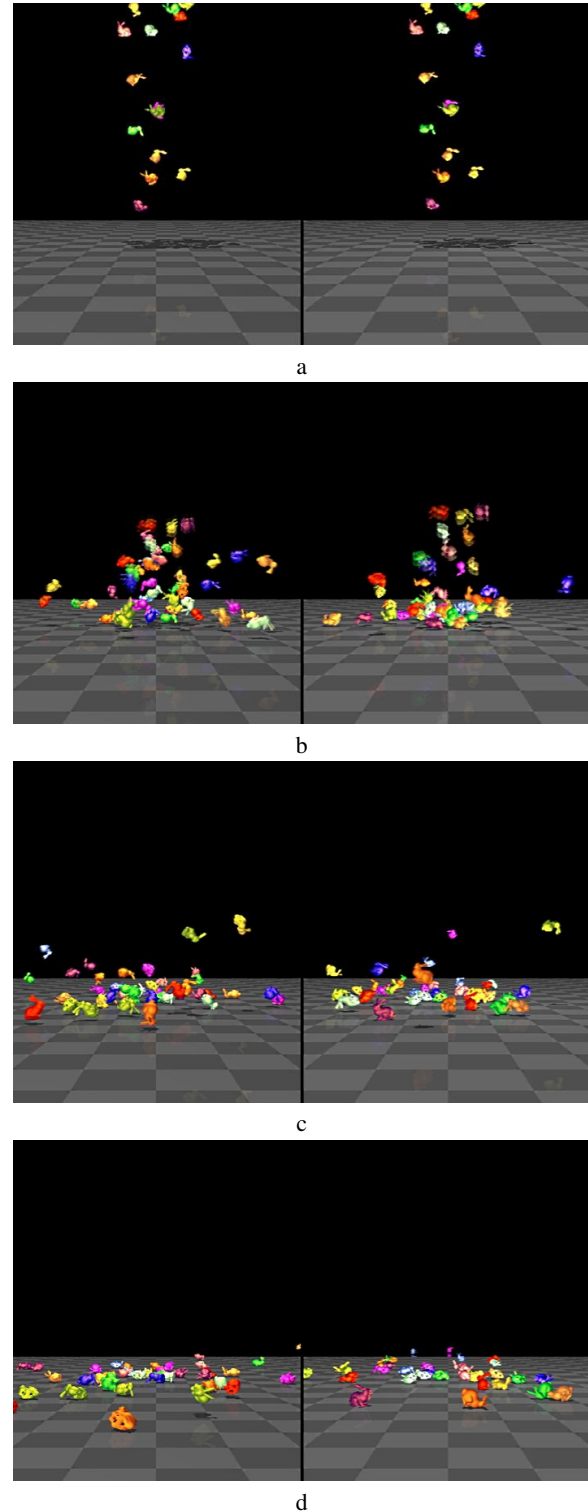


Figure 8: Simulation of 50 bunnies dropped to a plane. The left side of each image shows a statistical simulation, and the right side shows an impulse simulation. The statistical simulation is more than 50 times faster.