Time-critical collision handling for deformable modeling

Marc Gissler Ruediger Schmedding Matthias Teschner

Collision handling is a comparatively time-consuming task in dynamic simulations and the computational efficiency of collision handling techniques can vary significantly dependent on the spatial configuration of the environment. These issues have to be addressed in interactive simulations such as games or surgical simulators, where a pre-defined response time should be guaranteed for each simulation step.

We present a time-critical collision handling approach for deformable objects. The technique employs spatial subdivision for the detection of collisions and penetration depth information is computed to estimate penalty forces. Detection, penetration depth estimation and response are divided into atomic tasks. In case of an interruption, the algorithm basically resumes in the next time step. If collisions are not completely handled in one simulation step, the algorithm ensures that persistent collisions are handled in a subsequent simulation step. If an exact response cannot be computed in a given time frame, the algorithm efficiently approximates penalty forces for colliding points.

Experiments indicate that the proposed technique provides a physically-plausible collision handling in the case of incomplete or inconsistent collision information. User-defined limits for the computation time can be guaranteed with an efficiency gain of up to factor three.

Keywords: time-critical collision detection, time-critical collision response, deformable modeling

Introduction

In the context of virtual reality systems and interactive applications such as games and surgical simulators, there is a growing interest in the development of time-critical visualization and animation techniques that allow for the balancing of performance and quality [1]. Here, the challenge is to develop techniques that produce approximate, but also acceptable results considering a given time constraint. This is particularly challenging in the area of dynamic simulations, where the interplay of collision detection and response has to be considered for a physically-plausible collision handling.

Collision handling and the computation of the dynamics are commonly the two main tasks in physically-based animations. While the compu-

tation time for the dynamics is generally constant, the time for the collision handling can significantly vary. If objects are far away, the collision handling is very efficient. If, on the other hand, objects are in close proximity or colliding, the collision handling is rather expensive and may hinder interactive response rates of an application.

In order to meet a user-defined response time, time-critical methods tolerate an approximate, incomplete or inconsistent collision handling as long as the result is accepted by the user [2]. If, for example, adaptive representations are employed, the appropriate level-of-detail can be used to maintain a given response rate. This strategy is particularly useful in rigid-body simulations where bounding-volume hierarchies commonly accelerate the detection of collisions. In this case, an approximate collision detection can be realized by aborting a query at an arbitrary layer of the hierarchy.

However, since bounding-volume hierarchies are comparatively expensive to update, simple uniform grids are an efficient alternative for the collision detection of deformable objects. In this case, a time-critical collision handling approach has to work on the original object representation and a given response rate could be maintained by distributing the collision handling over multiple simulation steps. Our algorithm follows this idea.

Our contribution: We propose a time-critical collision handling algorithm for deformable objects. In contrast to approximate solutions that work on adaptive object representations, we propose to distribute the collision handling over multiple simulation steps.

In order to guarantee a given time constraint, collision detection, penetration depth estimation and response force computation can be interrupted at various points and postponed to subsequent time steps. The resulting inconsistencies and problems due to incomplete collision information are addressed by two approaches. First, if a collision is not handled in a simulation step, the algorithm ensures that a persistent collision is handled in a subsequent simulation step. Second, the algorithm efficiently predicts penalty forces, if the penetration depth and an exact response cannot be computed in time.

We show that the proposed scheme can be used to obtain physically-plausible results for an efficiency gain of up to factor three. We also illustrate that our technique is superior over naive solutions where collisions are simply handled in every second simulation step.

Related work

There exists a plethora of collision detection schemes for both rigid and deformable solids. Excellent surveys can be found in [3, 4, 5]. We first discuss collision detection schemes that are employed in collision handling approaches followed by a discussion of collision response schemes that specifically deal with collision information from interruptible collision detection approaches. Finally, we discuss the application of multi-resolution approaches to accelerate collision detection.

Time-critical collision detection

The first time-critical algorithm for collision detection was proposed by Hubbard [6, 7]. The author uses a bounding volume hierarchy (BVH) with spheres as bounding volumes (BV) to ap-

proximate polyhedral objects. Collisions are detected by traversing the hierarchies and refining the approximations. The refinement process can be interrupted at any level in the hierarchy to meet a time-constraint. If the bounding spheres still collide at this point, a collision response is invoked based on the approximation. The accuracy of the response can be estimated using information stored in the bounding spheres. In [8], the algorithm is extended to also handle deformable objects using a hybrid update method to update the BVs [9].

In [10, 11], the notion of average-distribution trees (ADB-trees) is introduced. They are an extension of conventional BVHs. In each BV, various characteristics about the average distribution of the set of primitives within the BV is stored. Based on this average distribution, an estimation of the probability that there exists a pair of intersecting primitives is derived. The approach allows for a numeric measure of the quality of the results. Thus, the quality of the collision detection can be reduced in a controlled way, while increasing the speed. Computing the additional information in the BVs is quite expensive. Therefore, the algorithm is best suited for rigid solids where these computations can be executed as a preprocessing step. Furthermore, only a general collision response can be derived from the BVs of the ADB-trees, since no information about the geometric orientation of the primitives is stored in the BVs.

In [12], perception-based prioritization is explored and several priority scheduling methods are proposed. Furthermore, information about colliding BVs is employed to approximate the response forces. The priority scheduling methods only consider one frame. Thus, collisions with low priority might not be handled. [13] propose stride-scheduling that prevent starving of low-priority collisions. They propose priority aging which increases the priority of an element in the queue with the time it spends waiting in the queue. Additional prioritization functions are described to minimize latencies and gaps between colliding objects, such as taking the masses and velocities of the objects into account. Furthermore, they are able to test at different frequencies independent of the fixed simulation frequency.

Collision response

Collision handling needs more than the pure detection of a collision, e.g. penetration depth and penetration direction is needed to compute a proper response [14, 15]. In the context of a time-critical collision handling scheme, a collision response scheme has to be robust in case of incomplete collision detection information. In [16], a method for the calculation of a consistent collision response in rigid body collision handling is proposed using information gathered with a time-critical collision detection scheme [7]. In [17], an extension of the method addresses the application of time-critical collision detection schemes on deformable models.

Multi-resolution approaches

Another category of methods that address the problem of consistent frame-rates is the category of adaptive meshes [18, 19, 20, 21]. Adaptive meshes store information on how to coarsen or refine a mesh to a certain resolution. Of course, a coarsened mesh can also be used to accelerate collision detection. The main challenge of

adaptive meshes is the transition between different resolutions.

In contrast to the multi-resolution and bounding volume hierarchy approaches, our approach directly works on the actual geometry, similar to [13]. Furthermore, it is applicable to deformable objects, which to our knowledge was not shown before.

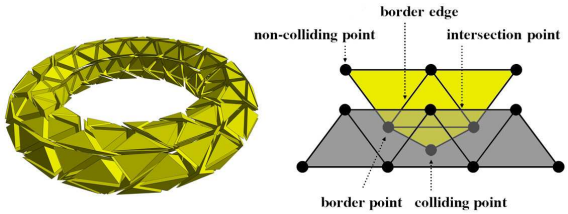

Figure 1: Left: Objects are represented as tetrahedral meshes that consist of points, edges, surface triangles and tetrahedrons. Right: In case of collisions, points are classified either as colliding or non-colliding. Further, border points, border edges and intersection points have to be determined for the penetration depth estimation.

Algorithm overview

In this section, we give an overview of our time-critical collision handling approach. We first give a short description of the object representation and clarify some terminology. We then recapitulate the basic principles of the collision detection, penetration depth computation and response force computation algorithms before we summarize the basic principles of our time-critical approach.

Object representation and terminology

In our simulation, we represent the volume of a deformable object by a tetrahedral mesh. Its surface is given by a triangular mesh (see Fig. 1 left). We need both representations in our collision handling scheme. If two objects penetrate each other, their points are classified as either colliding or non-colliding. A border edge pierces a surface triangle and connects a colliding with a non-colliding point. The edge-triangle intersection test returns an intersection point. And finally, border points are colliding points that share a border edge (see Fig. 1 right).

Collision handling overview

The collision handling scheme can be sectioned into three phases: collision detection, penetration depth estimation and collision response (see Fig. 2). Throughout the rest of this paper, we denote one execution of those three stages in sequence as one *collision cycle*. For collision detection, we employ a modified spatial hashing scheme in the spirit of [22]. It efficiently narrows down the candidates for a point-in-volume detection that finds collisions between any pair of objects. If collisions occur between objects, a collision response is computed in the second and third phase. We use a penalty-based response scheme that relates the magnitude of the response force per collided point to its penetration depth. A consistent penetration depth is estimated for colliding points by employing the approach described in [15].

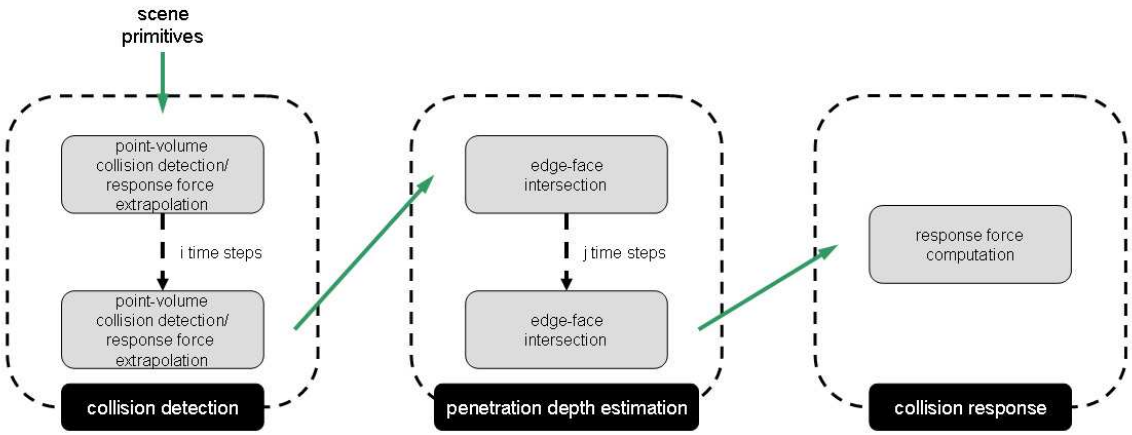

Figure 2: One collision cycle consists of three parts: collision detection, penetration depth estimation and collision response. Depending on the time constraint, execution of one collision cycle may take $i + j$ time steps.

Interruptible collision handling overview

In non-interruptible collision handling schemes, there is no limitation on the computation time for one collision cycle, whereas a time constraint is introduced in interruptible collision handling approaches. Collision detection and response are performed once per simulation time step in our framework. In our interruptible collision handling scheme, we follow the basic idea of processing granular tasks in the collision handling scheme until the time limit is reached and postponing unfinished tasks to the subsequent time step. We then continue collision handling with the processing of the first task in the list of unfinished tasks in the next time step. Thus, one collision cycle may be scattered over several time steps. We start a new collision cycle only after all tasks of the previous one were successfully performed. Furthermore, we propose an extrapolation scheme for the approximate computation of response forces. It is applied to colliding points in stage one, where collisions are detected. If time permits, these estimates are replaced by an accurate response force in stage three based on the penetration depth in stage two. One of the benefits is that we avoid inconsistencies in the application of response forces on points that are in contact for more than one time step.

Time-critical collision handling

In this section, we describe our time-critical collision handling scheme. We split the discussion into three parts. We first describe our time-critical collision detection scheme followed by the description of a time-critical penetration depth estimation. Finally, we describe how we maintain a plausible collision response in the case of incomplete collision information.

Time-critical collision detection

For collision detection, we employ the spatial hashing scheme from [22]. We choose this scheme, because it has been proven to be most suitable and very efficient in simulations where inter-object and intra-object collisions have to be found between deformable objects. Furthermore, all collision tests can be executed in a single query pass and the test may be interrupted in any and every iteration of the query pass. The scheme implicitly subdivides the possibly infinite simulation domain into a regular grid. Therefore, a hash function maps the three-dimensional cells of the grid into a one dimensional hash table of finite size. Primitives are hashed by finding the grid cells they intersect, mapping those grid cells into the hash table and leaving an imprint of the primitive in the corresponding hash table index. For collision detection, we perform a point-in-volume test. Therefore, all tetrahedrons are hashed and stored in the hash table in a first pass. In a second pass, point collisions are queried by hashing all points. If a point is associated with the same hash cell as one or more tetrahedrons, they might be in close proximity and an exact intersection test is performed for every point-tetrahedron pair. If all tetrahedrons and points are hashed and all collision tests are executed without interruption, the collision handling algorithm proceeds with the computation of a consistent penetration depth . On the other hand, if the time limit for collision handling is exceeded, the collision detection is interrupted and the collision response has to be provided with some estimations of the penetration depth .

Now, we discuss the locations that are feasible to interrupt the collision detection. Algorithm 1 shows the sequence of atomic tasks that are executed in the collision detection algorithm. There are only two locations at which the algorithm may be interrupted. Basically, it could be interrupted after any atomic task, but we believe that this is not preferable. First, we make the assumption that the upper time limit for the collision handling and the resolution of the scene geometry is chosen such that the collision handling could be processed within the time limit in case none of the objects in the scene is in close proximity of any other object. Otherwise, it would never be possible to handle collisions within the given time limit ever. Thus, we omit interruption tests after hashing one tetrahedron into the hash grid and after hashing all of them. The earliest feasible interruption location, therefore, is after querying a point for collision. In case a collision is found, an efficient collision response force estimation (see section "Response forces") is executed before interruption. We place the second spot for interruption after collision detection is completed. In the case of interruption, all remaining tasks are postponed to the next

Algorithm 1: Collision detection

Input: primitives of all objects in the scene
Output: point-tetrahedron pairs that are in collision with extrapolated response forces

```
foreach tetrahedron t do
    hash t and store in hash table
foreach point p do
    hash p into hash table
    foreach pair (p, t) in hash cell do
        query collision
    if collision then
        extrapolate response
    if time is up then interrupt
if time is up then interrupt
```

simulation step. Depending on the interruption spot, collision detection is resumed or we continue with the penetration depth estimation described in the next section. Note that hashing of the tetrahedrons is not repeated for the remaining collision tests to save computation time.

Time-critical consistent penetration depth

In order to resolve or avoid collisions, many collision handling schemes [23, 24] require some kind of interpenetration measure. We employ an efficient algorithm described in [15] to compute consistent penetration depths for colliding points. The algorithm works on the domain representation described in section "Algorithm overview", too.

The algorithm proceeds in four steps. In the first step, it searches for the border edges in the tetrahedral mesh of the colliding object. In the second step, the border edges are tested for intersection with the surface triangles of the mesh it is colliding with, returning the intersection points. Furthermore, the colliding points that share a border edge are marked as border points. The penetration depth of a border point is then computed by weighting the distances between the border point and all the adjacent intersection points in the third step. The fourth and final step propagates the penetration depths of the border points to colliding points that are no border points. For more details, see [15]. Algorithm 2 summarizes the main steps together with the interruptible spots.

Like in section "Time-critical collision detection", we now discuss the locations that are feasible to interrupt the penetration depth computation. First, we consider the computation of the intersection points. For efficiency, we again employ spatial hashing as in the collision detection algorithm to narrow down the pairs of triangles and edges that have to be tested for intersection. Thus, we hash and store all the surface triangles of the colliding meshes. Interruption is possible each time a surface triangle has been processed completely. Please note, that the triangle's current position is used for hashing to work on updated data. From here on, we process each colliding point on its own. First, we compute the penetration depth for each of the border points. For each adjacent edge of a border point, we test for intersection with a surface triangle using the spatial hashing scheme and then compute a weighted distance to the border points. As a result, we get the penetration depth of the border point and immediately derive a consistent response force.

Algorithm 2: Consistent penetration depth

```
Input: primitives of the colliding objects
Output: PDs for all colliding points
foreach surface triangle f do
    hash f and store in hash table
    if time is up then interrupt
foreach colliding point p do
    foreach adjacent edge e of p do
        hash e into hash table
        foreach pair (f, e) in hash cell do
            query intersection
            if intersection then
                query intersection point
                mark p as border point
    if p is border point then
        compute penetration depth for p
    if time is up then interrupt
if time is up then interrupt
while PD is not computed for all points p do
    propagate PD to adjacent colliding points
    if time is up then interrupt
```

This force overrides the estimated response force computed after the collision detection (see section "Time-critical collision detection" and section "Response forces"). After the penetration depths and response forces are computed for all the border points, we can propagate the penetration depth to all adjacent colliding points that are no border points and then continue layer by layer. We may interrupt every time a colliding point has been completely handled (see algorithm 2).

Response forces

Now, we discuss the computation of the response forces for our interruptible collision handling scheme. For efficiency, we employ a simple penalty-based method that relates the magnitude of the response force per collided point to its penetration depth [15] using a response constant c_r: $f_p = d_p * \mathbf{d}_p * c_r$, where d_p is the penetration depth for a colliding point p and $\mathbf{d}_p$ its penetration direction. Additionally, some friction and damping forces may be applied to the response forces.

In our time-critical collision handling approach, we have to consider two cases. First, if the collision handling can be completed within the given time limit, response forces are computed for all collisions in stage one and three in every time step. In stage one, response forces are efficiently approximated. If time permits, these estimates are replaced by an exact computation in stage three based on the penetration depths computed in stage two. Thus, for all detected collisions, a response force is either approximated or accurately computed.

In the extrapolation scheme, we consider the response forces of a point p in contact computed in two previous time steps. A good choice is to use response forces based on the penetration depths. Otherwise, the input of the extrapolation scheme would be forces that were extrapolated themselves. We denote $\mathbf{f}_{c-1}^{response}$ and $\mathbf{f}_{c-2}^{response}$ to be those response forces from the last two collision cycles. We linearly interpolate these two forces over the time interval that spans the last collision cycle to get the force difference between two consecutive time steps: $\mathbf{f}_{diff}^{response} = \frac{\mathbf{f}_{c-1}^{response} - \mathbf{f}_{c-2}^{response}}{n}$, where n is the number of time steps of the last collision cycle. Here, we assume that the current collision cycle requires the same amount of time steps to be completed. As was mentioned previously, the response forces could be composed of more than just the penalty forces based on the penetration depth, e.g. friction and damping forces may be added as well. However, they are implicitly considered in the force extrapolation, since they are included in the force difference. Furthermore, there is one special case we have to account for. If the response force is already decreasing, it is possible that the estimated force we add may lead to a response force that changes its direction . In this case, we simply cancel it out the and compute the response force based on the penetration depth in stage three. Application of estimated response forces for points that are in collision across many collision cycles and simulation time steps is crucial for the robustness of the simulation. For example, considerable jitter can be avoided in stacking scenarios like the one described in the results section.

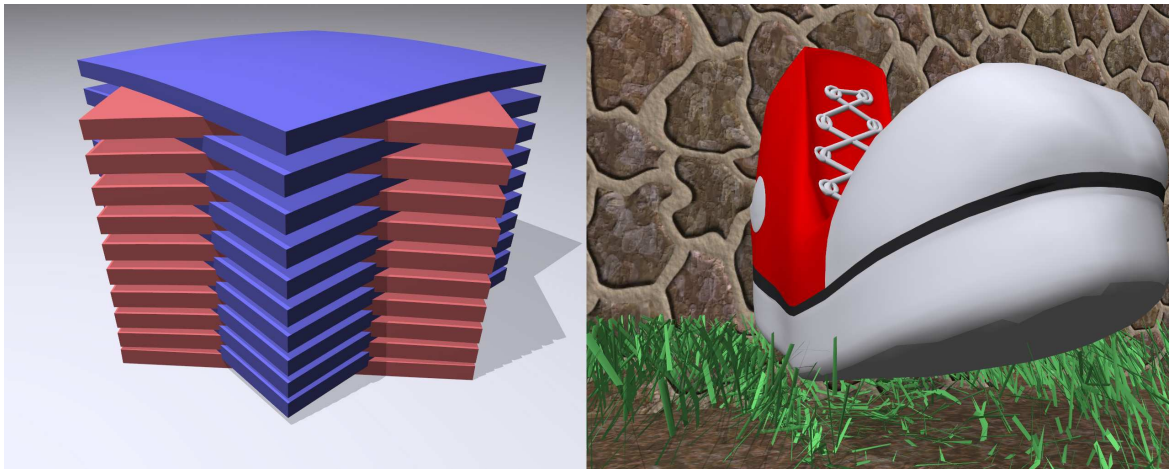

Figure 3: The test scenarios demonstrate various effects in collision handling: Resting contact (left) and bouncing contact (right).

Results

We evaluate our interruptible collision handling approach using a set of test scenarios. We simulate the scenarios with and without time constraints for the collision handling.

In the first scenario, 20 plates are stacked up to show the impact of time constraints on resting contact (see Fig. 3 left). In the second scenario, a sneaker falls on a piece of grass to show bouncing contact (see Fig. 3 right). Using these scenarios, we discuss various properties and challenges of interruptible collision handling for deformable modeling: meeting time constraints, latency and speedup, and plausibility of the visualized simulation results. The approach has been integrated into a deformable modeling framework based on the Finite Element Method for tetrahedrons [25] to exemplify its applicability on deformable objects. All timings have been performed on an Intel Core 2 PC, 2.13 GHz with 2 GB of memory. The code is not parallelized.

Meeting time constraints

In order to reduce the percentage of collision handling in the overall computation time of the simulation, one naive solution would be to perform collision handling only every second or more simulation step. However, this approach leads to inconsistent collision response forces and resting contact is hard to maintain (see the accompanied video). With the introduction of the interruptible spots in the collision handling scheme and the force extrapolation scheme, it is possible to both maintain a target frame rate for collision handling and maintain plausible collision behavior. The charts in figure 4 show the computation times for the collision handling with and without time constraints for the test scenarios. They demonstrate that the time constraints are met with a granularity of less than five percent of the assigned computation time. Despite the time-constraint, the simulation still computes plausible resting and bouncing contacts, which is further discussed in the next section (see also the

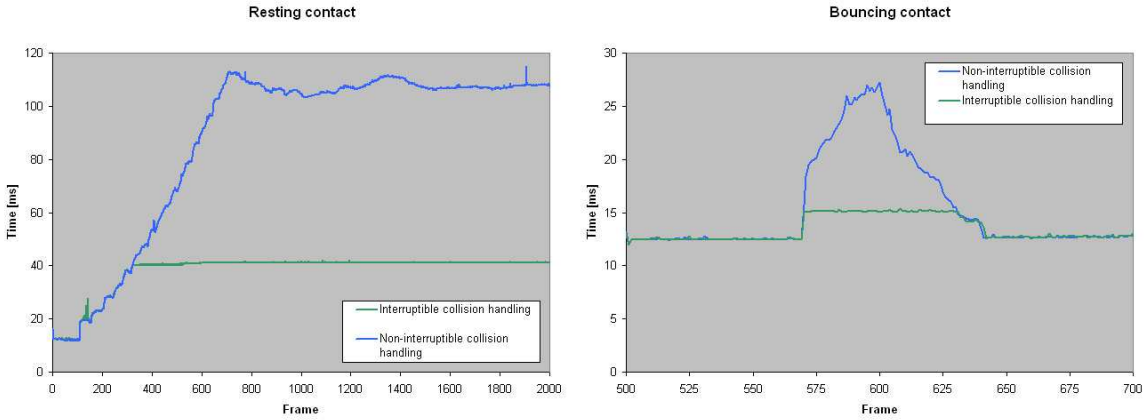

Figure 4: Computation time of the collision handling scheme for the resting contact scene (left) and bouncing contact scene (right) with and without interruptions. In the case of interruption, the user-defined time constraint is set 40 and 15 milliseconds, respectively. Time constraints are met with a granularity of less than five percent of the given computation time.

accompanied video). Please note that the time constraint should be chosen appropriate to the test scenarios. It should be possible to hash and store at least all tetrahedrons and hash all points located in the scene. Otherwise, the collision detection cycle would always take more than one simulation time step, even in a collision-free simulation environment.

Latency

In time steps with high amounts of collisions, a collision cycle may span several simulation time steps due to the user-specified time constraint. This introduces a delayed collision response for newly colliding objects (i. e. points that are not already in contact and where the response force extrapolation cannot be applied) and may increase the interpenetration. On the other hand, response forces may be applied onto objects that are in contact for too long using the force extrapolation. Furthermore, objects might move right through each other, while the response force are just computed in one of the later frames of the collision cycle. These phenomena are already inherent in the penalty-based response force computation approaches. Thus, the question is how much latency may be introduced by the interruptible collision handling approach without amplifying the described effects too much. Our test scenarios show that the answer depends on the relative velocities of colliding objects. For example, in the stacking scenario, a latency of up to four frames still produces robust results, whereas in the massive scene with the very fast moving heavy sphere, a latency of up to two frames guarantees the execution of response forces in time, i. e. no torus moves through the sphere and the simulation keeps stable. We believe that prioritizing objects with high velocities in the collision handling could increase the acceptable latency in the second scenario as well. Depending on the scenario and the chosen time constraint, interrupt-

ible collision handling saves up to two third of the computation time for collision handling per frame when compared to the same scenario without a time constraint. For example, in frame 700 of the stacking scenario, collision handling takes 120 milliseconds in the standard collision handling scheme and only 40 milliseconds using our interruptible approach. In the next subsection, we will discuss the topic of plausibility in more detail.

Plausibility

It is obvious and inevitable that any interruptible collision handling approach trades accuracy and realism for efficiency. The outcome of such a trade-off is a simulation of lower quality. Luckily, studies have shown that people have certain weaknesses in detecting anomalies in dynamic simulations [26]. Thus, simulations of possibly lower quality such as the ones using an interruptible collision handling may still look plausible and believable to the viewer. To check this assumption, we designed our scenarios and chose the time constraint for the interruptible scheme such that we avoid obvious artifacts like objects moving through each other, severe interpenetrations or unstable behavior due to very large response forces. We then recorded sequences with and without the time constraint and presented them to people from inside and outside the research group. The task was to name the sequences that were recorded with the time constraints. This quick experiment revealed that it seems difficult to label the sequences correctly. However, it is also difficult to specify the size of a collision cycle that avoids obvious artifacts.

Discussion: O'Sullivan et al. [27] present a set of psychophysical experiments that establish thresholds for human sensitivity to dynamic anomalies, including angular, momentum and spatio-temporal distortions applied to simple animations depicting the elastic collision of two rigid objects. In previous research [2], they also show that the latency between expected and displayed time of collision response impacts believability. The longer the collision handling approach spends processing collisions and the longer the delay that is thus generated, the less believable the resulting collisions will be. It is hard to judge how some of these findings carry over to deformable objects. Non-rigid objects deform on collision either elastically or plastically and therefore absorb parts of the collision energy. As a result, they may stay longer in contact than rigid bodies during an elastic collision. A quick survey among 20 people from inside and outside the research group revealed some interesting aspects. First, the expected time of collision response varies considerably. Participants described it difficult to predict the amount of deformation and the duration of contact for pairs of colliding objects. Second, it seems to be difficult to predict the trajectory of colliding objects after collision and contact is resolved. Of course, these findings are in no way the results of a qualitative examination like in [2] and it would be interesting to extend their findings to the more complex animations of the collision of two deformable objects.

Conclusion

We have presented an interruptible collision handling approach for deformable objects. Our approach works on the original object representa-

tion and a given response rate could be maintained by distributing the collision handling over multiple simulation steps. We have shown and discussed the various points where it is reasonable to interrupt the collision handling. Furthermore, we have shown, how a persistent collision is handled in a subsequent simulation step and how a response force can be predicted until the penetration depth and an exact response force can be computed within the given time constraints. Experiments have shown that the proposed scheme can can be used to obtain physically-plausible results for an efficiency gain of up to factor three. In future research, would like to evaluate the visual fidelity of the animation of deformable objects in a more qualitative experimental investigation.

Acknowledgments

We would like to thank Roman Engels for many fruitful discussions. Furthermore, this work has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8.

References

[1] A. van Dam. VR as a forcing function: Software implications of a new paradigm. In *IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 5–8, 1993.

[2] C. O'Sullivan and J. Dingliana. Collisions and perception. *ACM Trans. Graph.*, 20(3):151–168, 2001.

[3] M. Lin and D. Manocha. Collision and proximity queries. In *Handbook of Discrete and Computational Geometry*, 2003.

[4] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24:61–81, 2004.

[5] C. Fares and Y. Hammam. Collision detection for rigid bodies: A state of the art review. In *GraphiCon 2005*, 2005.

[6] P. M. Hubbard. Interactive collision detection. In *IEEE Symp. Research Frontiers in Virtual Reality*, pages 24–31, 1993.

[7] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15(3):179–210, 1996.

[8] J. Dequidt, D. Marchal, and L. Grisoni. Time-critical animation of deformable solids: Collision detection and deformable objects. *Comput. Animat. Virtual Worlds*, 16(3-4):177–187, 2005.

[9] T. Larsson and T. Akenine-Moeller. Collision detection for continuously deforming bodies. In *Eurographics*, pages 325 – 333, 2001.

[10] J. Klein and G. Zachmann. Time-critical collision detection using an average-case approach. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 22–31, New York, NY, USA, 2003. ACM.

[11] J. Klein and G. Zachmann. ADB-Trees: Controlling the error of time-critical collision detection. In *Proceedings of the 8th International Fall Workshop Vision, Modeling, and Visualization 2003 (VMV 2003)*, pages 37–45, 2003.

[12] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees. In *Spring Conference on Computer Graphics*, pages 83–92, 1999.

[13] D. S. Coming and O. G. Staadt. Stride scheduling for time-critical collision detection. In *VRST*

'07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, pages 241–242, New York, NY, USA, 2007. ACM.

[14] S. Cameron. Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. *IEEE International Conference on Robotics and Automation*, 4:3112–3117, 1997.

[15] B. Heidelberger, M. Teschner, R. Keiser, M. Müller, and M. Gross. Consistent penetration depth estimation for deformable collision response. In *Proceedings of Vision, Modeling, Visualization VMV04*, pages 339–346, 2004.

[16] J. Dingliana. Graceful degradation of collision handling in physically based animation. In *Proc. Eurographics 2000*, pages 239–247, 2000.

[17] O'Sullivan C. Mendoza, C. Towards time-critical collision detection for deformable objects based on reduced models. In *Proceedings of the Conference on Computer Animation and Social Agents (Posters)*, pages 1–2, 2005.

[18] O. Staadt and M. Gross. Progressive tetrahedralizations. In *Proceedings of the Conference on Visualization*, pages 397–402, 1998.

[19] E. Grinspun, P. Krysl, and P. Schröder. Charms: a simple framework for adaptive simulation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290, New York, NY, USA, 2002. ACM.

[20] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 31–36, New York, NY, USA, 2001. ACM.

[21] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph.*, 25(3):820–825, 2006.

[22] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings Vision, Modeling, Visualization (VMV'03)*, pages 47–54, 2003.

[23] K. Hauser, C. Shen, and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Graphics Interface*, pages 247–255, 2003.

[24] J. Spillmann, M. Becker, and M. Teschner. Non-iterative computation of contact forces for deformable objects. *Journal of WSCG*, 15(1–3):33–40, 2007.

[25] M. Müller and M. Gross. Interactive virtual materials. In *GI '04: Proceedings of Graphics Interface*, pages 239–246, 2004.

[26] J. Clement. Students' preconceptions in introductory mechanics. *American Journal of Physics*, 50(1):66–71, 1982.

[27] C. O'Sullivan, J. Dingliana, T. Giang, and M. K. Kaiser. Evaluating the visual fidelity of physically based animations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 527–536, New York, NY, USA, 2003. ACM.