To appear in ACM TOG 34(4).

Air Meshes for Robust Collision Handling

Matthias Müller Nuttapong Chentanez Tae-Yong Kim Miles Macklin
NVIDIA

Figure 1: *A dancer with a multi-layered skirt. Our method robustly simulates the complex interaction of the layers and is able to smoothly recover from any entangled state.*

Abstract

We propose a new method for both collision detection and collision response geared towards handling complex deformable objects in close contact. Our method does not miss collision events between time steps and solves the challenging problem of untangling automatically and robustly. It is conceptually simple and straight forward to parallelize due to the regularity of the algorithm.

The main idea is to tessellate the air between objects once before the simulation and by considering one unilateral constraint per element that prevents its inversion during the simulation. If large relative rotations and translations are present in the simulation, an additional dynamic mesh optimization step is needed to prevent mesh locking. This step is fast in 2D and allows the simulation of arbitrary scenes. Because mesh optimization is expensive in 3D, however, the method is best suited for the subclass of 3D scenarios in which relative motions are limited. This subclass contains two important problems, namely the simulation of multi-layered clothing and tissue on animated characters.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

Keywords: collision detection, mesh optimization

1 Introduction

Both collision detection and collision handling are challenging problems and active areas of research in computer graphics [Teschner et al. 2005], [Yoon et al. 2010]. Existing methods for collision detection typically use spatial acceleration structures such as uniform grids or bounding volume hierarchies to identify overlapping primitives. For thin or fast moving objects, the swept volume over a time step has to be considered, rather than the static volume of the objects not to miss any collision events. For complex objects in close proximity such as layers of clothing, this step is expensive and likely the bottleneck of the simulation.

Once pairs of overlapping primitives have been identified, appropriate response forces have to be applied in order to resolve the collisions. For thin structures such as cloth or deeply penetrating volumetric objects, finding the correct directions of the response forces is challenging. Properly recovering from an entangled state is a global problem so instead of analyzing the collision pairs individually, entire objects have to be considered [Wicke et al. 2006]. One way to solve this problem is to avoid entangled states all together via exact collision handling [Brochu et al. 2012]. These methods tend to be computationally expensive. Also, kinematic objects such as characters often force clothing into penetrating configuration so avoidance is not possible in those cases [Baraff et al. 2003].

In this project we were specifically interested in collision handling for characters with simulated multi-layered clothing, simulated tissue and the interaction of the two (see Figure 1). In this scenario it is often the case that entangled states are unavoidable at certain points in time. Examples are when a character sits or lays down or when the clothing is trapped below the armpits. Also, entangled states may result from the fact that the time for resolving collisions is limited. Typically, temporary collisions are tolerable as long as they are resolved over time. On the other hand, persistent entangled states are unacceptable.

The most important feature of our new method is therefore its ability to recover smoothly from any entangled state. This is a key piece for bringing tissue and multilayered clothing simulation to games. Our method also handles collision detection and collision response simultaneously and considers continuous collisions. The algorithm is conceptually simple to understand and to implement.

Our contributions are:

- The idea to triangulate the air between objects for robust collision handling.

- Avoiding collisions by forcing the volumes of the elements of the triangulation to be positive via a unilateral constraint.
- Using mesh optimization to allow objects to freely move relative to each other.

2 Related Work

In this section we cover the most closely related previous work on collision detection and collision response and refer the reader to [Teschner et al. 2005], [Yoon et al. 2010] for more comprehensive surveys on these topics.

2.1 Collision Detection for Deformable Objects

To detect inter-object and self collisions of deformable objects such as cloth or tissue, all pairs of primitives, e.g. triangles or tetrahedra have to be tested for overlaps. Typically, this task is the most expensive part of the collision handling pipeline due to the high number of potential pairs and the fact that the spatial relationship between primitives changes every frame when the objects deform. Therefore various acceleration schemes have been proposed to efficiently cull unnecessary tests. Examples are spatial hashing [Teschner et al. 2003], chromatic decomposition of orthogonal triangle sets [Govindaraju et al. 2005], connectivity based culling [Tang et al. 2008], separating axis tests [Tang et al. 2011], radial view based culling [Wong et al. 2013], [Wong and Cheng 2014], [Wong et al. 2014], simulation space based culling [Barbič and James 2010], [Zheng and James 2012], the use of representative primitives [Curtis et al. 2008], rasterization of objects using GPUs [Govindaraju et al. 2003], [Heidelberger et al. 2004], [Faure et al. 2008], [Wang et al. 2012] and various hierarchical methods [Mezger et al. 2003], [Schvartzman et al. 2010], [Schvartzman et al. 2009].

2.2 Continuous Collision Detection

For deformable objects, missing a single collision can create an entangled state from which it is hard to recover later. Bridson et al. [2002] presented a robust collision handling pipeline using continuous collision detection (CCD) between triangles of cloth meshes. Various methods have been proposed to reduce the cost of CCD, such as conservative advancement [Zhang and Kim 2012], non-penetration filters [Tang et al. 2010] and bounding volume hierarchies [Hutter and Fuhrmann 2007], [Wong et al. 2010]. Although analytically correct, these methods can produce false negatives due to numerical inaccuracies. This problem has been addressed by Brochu et al. [2012] who use exact and interval arithmetic for CCD tests. Wang et al. [2014] presented a set of criteria that automates the selection of numerical thresholds and Tang et al. [2014] used Boolean collision queries based on Bernstein sign classification.

2.3 Volume Based Approaches

Our method is quite different from all these approaches in that we detect collisions by testing whether the space between objects has inverted during the time step. Sifakis et al. [2008] et al. also work with the inter object space but instead of detecting inversions in a persistent mesh, they conserve the volume of temporal elements which are re-initialized at the beginning of each time step. Instead of considering the volume between objects, Allard et al. [2010] derive contact forces from the overlapping volume of objects. Levin at al. [2011] use an Eulerian approach to handle collisions between solids. In this setting, the air volume between objects is modeled as well but in contrast to our approach on a regular fixed background grid. Here, collisions are handled in the traditional way because the cells keep their shape and do not invert.

The idea of considering the volume between objects for collision handling has been proposed outside of the graphics community. Agarwal et al. [2000] maintain a kinetic data structure they call *pseudo-triangulation* for collision detection between 2D objects. In contrast to our simple edge flip approach, these authors are able to prove that only true collisions are detected. However, due to its complexity and restriction to 2D, the method is not well suited for the real time simulation of complex objects. Pagano et al. [2008] also discusses the idea of preventing space from inverting for collision detection but mostly in the continuous case without discussing collision recovery and most mesh based aspects. Their method is also restricted to 2D.

2.4 Untangling and History Based Handling

Sometimes, the deformable objects are forced into entangled states. In clothing simulations this can happen below the armpit of a kinematically animated character or when the character sits or lays down. This situation cannot be prevented by CCD algorithms. To address this problem, Baraff et al. [2003] introduced a cloth *untangling* scheme that analyzes intersecting cloth geometry and generates response forces to gracefully untangle the cloth over time. The method was extended by Wicke et al. [2006] to handle meshes with boundaries as well and by Volino et al. [2006] who introduced a contour minimization scheme for untangling.

A different approach to the untangling problem is to pull the configuration to an earlier, collision free state. Such a history-based method was proposed by Selle et al. [2008] to decide whether attraction or repulsion forces are needed to resolve collision pairs. Our method falls into the same category. While Selle et al. re-initialize the memory of the untangled state every time step and only use it during the per-time step iterative scheme, we initialize it once at the beginning of the simulation and evolve it over the entire simulation. In our case the memory consists of the mutual spatial positions of primitives. A more specialized memory specifically designed for multi-layered clothing was proposed by Perez et al. [1999] who pull each layer toward an iso-surface defined around an animated skeleton. Wong et al. [2004] pull the cloth vertices in a given layer towards a position given by barycentric interpolation plus some fixed normal offset from the triangles in the previous layer.

A common method to simulate complex geometry robustly is to embed it in a surrounding tetrahedral mesh [Müller and Gross 2004]. The two main differences to our idea is that in our case, the vertices of the air mesh coincide with the vertices of the simulated visual geometry and that the air elements are completely inactive as long as their volumes are positive. In the accompanying video we show how significantly different the skirt in Figure 1 behaves when full volume conservation is turned on.

2.5 Inversion Handling and Mesh Improvement

The basic idea behind our method is to track the air mesh between the objects. Collision handling happens automatically by performing two tasks, namely inversion handling and mesh improvement. Treating inverted elements in connection with the finite element based simulation of deformable solids has been addressed by [Irving et al. 2004], [Irving et al. 2007] [Civit-Flores and Susn 2014] and [Stomakhin et al. 2012]. The basic idea is to determine the smallest negative eigenvalue of the strain tensor and to apply restoring forces in a direction derived from the corresponding eigenvector. Since we use Position Based Dynamics, we simply add a unilateral constraint forcing the volume of each element to be positive [Müller et al. 2007].

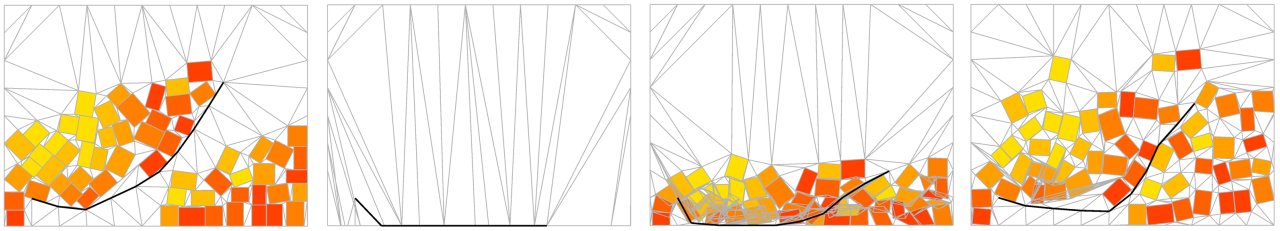

Figure 2: *A 2D scene composed of 60 boxes and a rope. The air mesh (white) stably prevents the boxes from passing through the one dimensional rope. After compressing the entire scene onto the x-axis, the objects and their relative positions are recovered.*

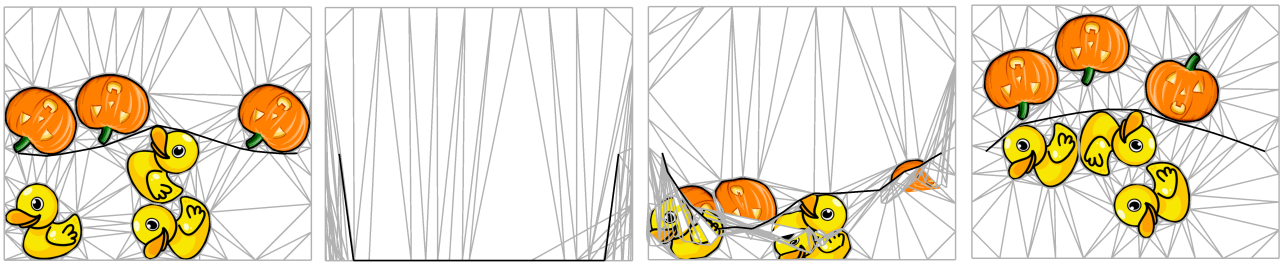

Figure 3: *General objects in 2D. Again, the air mesh keeps the objects on the correct side of the rope even after the scene is compressed into a line.*

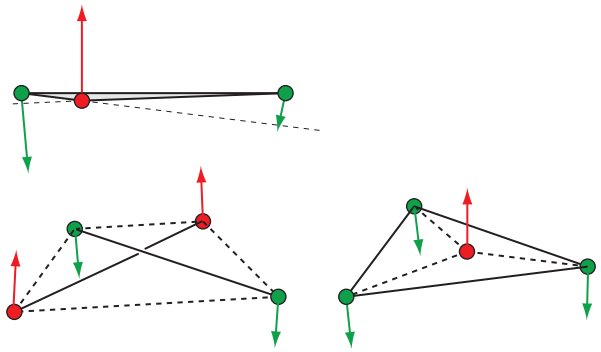

Figure 4: *Gradient vectors of the volume constraint on air elements for all non-locking two body collision cases in 2D (triangle, top row) and 3D (tetrahedra, bottom row).*

There is a large body of work on tetrahedral mesh improvement. Fritag and Ollivier-Gooch [1997] utilized smoothing and several topological operations such as $2 \rightarrow 3$ and $3 \rightarrow 2$ flips as well as general edge removal to improve mesh quality. Klingner and Shewchuk [2007] proposed a vertex insertion operation and utilizing various heuristics that yield better mesh quality. This approach was further refined by Wicke et al. [2010] to simulate large elasto-plastic deformation of solids. We use general edge removals and multi-face removals described by Shewchuck [2002].

3 The Method

Our method consists of three steps. First we tessellate the air between objects as a pre-processing step before the simulation. During the simulation we consider unilateral (inequality) constraints that prevent the air elements from inverting. Third, to avoid artifacts in the case of large rotations we perform a dynamic mesh optimization step during the simulation (see Figure 6). Let us now look at these three steps in more detail.

3.1 Tessellation of the Space Between Objects

As a pre-processing step, the air between objects has to be tessellated. In our simple generic examples (Figures 2, 5 and 6) we generated the air elements procedurally. In the general case, a conforming tessellation of the space between objects is needed that respects the surfaces of the simulated meshes. Since this step has to be executed only once before the simulation starts, it is not time critical. There are existing packages that can be used for this task such as *TetGen* by Si [2015].

3.2 Per Element Constraints

During the simulation, the air elements are passive as long as their volume is positive. Only inverted elements exert restoring forces to make their volume zero. Such unilateral constrains can be formulated in force or impulse based systems. Since we use Position Based Dynamics (PBD) [Müller et al. 2007] in our simulations, we simply consider a unilateral volume constraint per air element

$$C_{\text{air}} = |(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)| \geq 0 \quad (1)$$

$$C_{\text{air}} = \det[\mathbf{p}_2 - \mathbf{p}_1, \mathbf{p}_3 - \mathbf{p}_1, \mathbf{p}_4 - \mathbf{p}_1] \geq 0 \quad (2)$$

in the 2D and 3D case, respectively, where $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{p}_4$ are the particles adjacent to the element. The set of non-locking air element inversions with two adjacent bodies shown in Figure 4 corresponds to the general primitive collisions cases, i.e. vertex-edge collisions in 2D and edge-edge and face-vertex collisions in 3D. Momentum is conserved by construction of the PBD update. For instance, the green particle that is closer to the red particle in the 2D case receives a larger positional update. Instead of using the gradients of the volume constraint, it is also possible to compute the normals directly from the colliding objects if references to them are stored in the particles.

Our parallel Jacobi-type unified solver allows the fine-grained, interleaved processing of various constraints. Theoretically, the collision kernel can be placed at any position in the solver loop. However, to give collision handling priority over elasticity, we execute

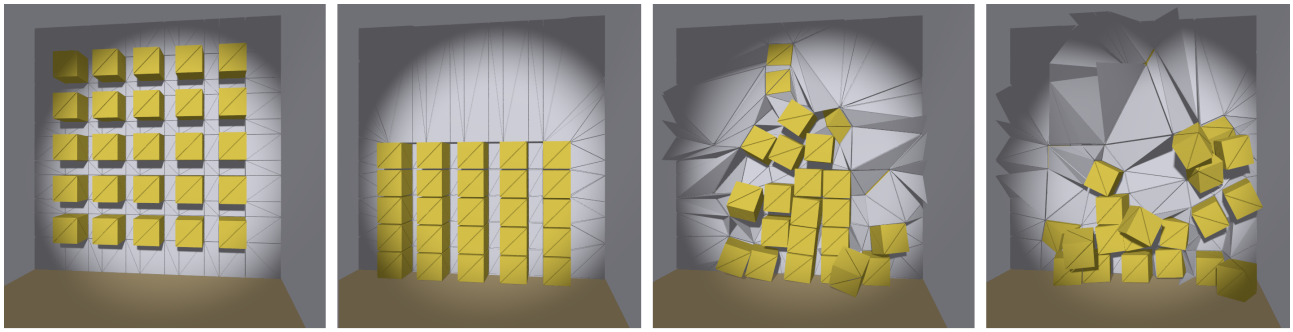

Figure 5: *A cut through a 3D scene with object tetrahedra in yellow and air tetrahedra in gray. Top left: the initial state. Top right: when gravity is turned on, the boxes drop to the ground and form stable stacks. Bottom row: mesh optimization lets the objects move freely w.r.t. each other.*

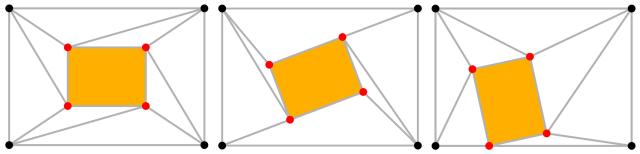

Figure 6: *Left: A simple 2D case in which a dynamic orange box is surrounded by a rectangular border. By preventing the white triangles to invert, the box collides correctly against the walls. Middle: When the relative rotation of the box w.r.t. border becomes too large, the air triangle lock even though the box does not collide. Right: Optimizing the air mesh via flips solves the problem allowing the box to move freely.*

it last within each iteration possibly multiple times resulting in a higher overall iteration count for collision handling.

3.3 Mesh Optimization

As the experiment in Figure 6 shows, air elements might get inverted even if there are no collisions between objects. This can happen when the objects undergo large relative rotations or translations. In such cases, the air mesh locks, preventing the objects to rotate freely. The kinetic pseudo triangulation proposed by Agarwal et al. [2000] solves this problem and only reports true collisions, a fact that can be formally proved. However, as mentioned earlier, this data structure and its update algorithm are quite expensive. We found that edge flips alone prevent locking very effectively. In the 2D case shown in Figures 2 and 3 we perform edge flips whenever they improve the minimum quality of the adjacent triangles. We use

$$q_{\text{triangle}} = \frac{4}{\sqrt{3}} \frac{A}{l_1^2 + l_2^2 + l_3^2} \tag{3}$$

as the quality measure, where A is the area of the triangle and l_1, l_2 and l_3 the lengths of its edges. This measure is 1 for equilateral triangles. Even though there is no theoretical guarantee that locking cannot occur with edge flips alone, it is hard to spot any locking artifacts in the scenes even with a large number of objects and complex interactions as the accompanying video shows.

It was clear that edge flips alone would let the box in Figure 6 rotate freely. That distance constraints on the object's edges and simple air constraints with edge flips would yield a perfectly looking 2D soft body simulation with a large number of freely moving objects and ropes, continuous collision detection and stable stacking as shown in Figures 2 and 3 came as a surprise however.

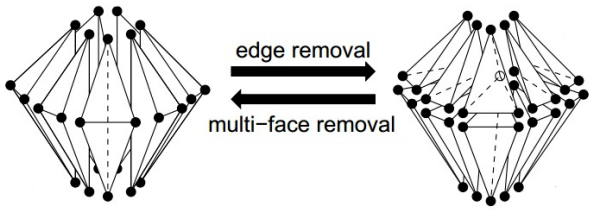

Figure 7: *Edge removal and multi-face removal operations on a tetrahedral mesh. Image courtesy of Jonathan Shewchuk.*

It is important to note that not all flips are allowed. A flip of an edge is only allowed if both adjacent triangles belong to the air mesh and the edge itself is not part of a rope. Otherwise, object would be able to freely penetrate.

As the experiment in Figure 5 shows, the same concept works in 3D too although optimizing a tetrahedral mesh is a significantly more complex problem than optimizing a triangle mesh. Instead of simple edge flips, a variety of optimization operations have been proposed in the literature. We use *edge removals* and *multi-face removals* [Shewchuk 2002] which are depicted in Figure 7 and the quality measure

$$q_{\text{tetrehedron}} = \frac{12}{\sqrt{2}} \frac{V}{l_{\text{rms}}^3}, \quad l_{\text{rms}} = \sqrt{\frac{l_1^2 + l_2^2 + l_3^2 + l_4^2 + l_5^2 + l_6^2}{6}}, \tag{4}$$

where l_{rms} is the root mean square of the edge lengths l_1, l_2, l_3, l_4, l_5 and l_6 and V the volume of the tetrahedron. This measure is 1 for regular tetrahedra. Similar to the 2D case, optimization operations are only allowed if all removed tetrahedral faces are adjacent to air cells only and not part of a piece of cloth.

4 Discussion

In this section we will discuss the properties of our approach.

4.1 Memory

An important aspect of our method is the fact that the air mesh has a memory of the relative positions of the objects. This information is stored in the orientation of the air elements and not lost even if the elements are made completely flat.

In contrast to the approach of Selle et al. [2008] where the history is re-initialized at the beginning of each time step, the information stored in the air mesh in initialized only once at the beginning of the simulation and evolves consistently over time across mesh optimization operations. Unlike [Selle et al. 2008], this information allows the correct recovery from entangled states.

Inspired by the experiment of [Müller et al. 2005] in which a single duck recovers from complete flatness, we started with the main scene of Grimm's fairy tale *The Town Musicians of Bremen* [Grimm and Grimm 1812] in which four animals stand on top of each other (see Figure 8). After compressing them to a state in which all y-values are zero – which is not part of the story –, they not only recover individually but their vertical order is preserved as well.

Figures 2, 3 shows the equivalent experiment in 2D. In the second image, all y-values were forced to zero. The following screen shots show how the entire scene recovers while the objects pop up in the right order and on the correct side of the rope.

A more practical and more important case is cloth untangling as in Figure 10. Our approach not only detects entangled states, it also allows the smooth, non disturbing recovery from arbitrary entangled states over a number of time steps. We are not aware of existing methods with this ability.

4.2 Turning Optimization Off

Due to the complexity of 3D mesh optimization, air meshes are not the best choice for handling the typical benchmark scene of freely moving and rotating objects falling into a bowl. Those types of scenes are not the common case in games or film, however.

Fortunately there is a subset of problems on which our method works very well even without the optimization step, and which is of great importance in games and film, namely clothing and tissue simulation as shown in Figures 1 and 9. These are also the most difficult cases for traditional collision handling approaches. The reason why the omission of the optimization step does not cause disturbing artifacts is that layers of clothing or tissue mostly move perpendicular to their surface without large rotations.

The key observation that motivates the limited application or even omission of the optimization step is that the orientation constraints alone provide conservative collision handling in the sense that they still guarantee a collision free state which is essential for clothing and tissue at the cost of visual artifacts which are minimal as our examples show. This is in contrast to existing methods in which limited iterations cause overlapping states.

Besides optimizing at each time step and turning optimization off completely, it is also possible to perform mesh optimization only every nth time step or on an alternating subset of the elements to distribute the load over multiple time steps. There is another way to treat accuracy for speed, namely playing with the number of iterations over the air constraints. Low iteration counts may leave intersection unresolved in fast motions but they always recover correctly when the motion slows down again.

4.3 Hybrid Collision Handling

To maintain an epsilon distance between objects and to handle static and dynamic friction we perform a second collision handling pass after the air mesh projection. Here, we detect pairs of primitives within and epsilon distance by walking the air mesh and apply the regular collision handling constraints for separation and friction described in [Macklin et al. 2014].

Such a hybrid approach can also be used to reduce the number of air elements. For detailed collision meshes, the number of air elements can become quite large. In this case, our method can be used as a broad collision phase which makes sure that objects are on the correct side of each other after the projection using coarse collision representations. A second pass of traditional collision handling makes sure that the smaller features do not penetrate. Due to the reduced number of air elements, this approach would have the potential to make our approach practical for arbitrary scenes in 3D. We have not tested the second type of a hybrid method, this is part of our future work.

5 Results

In all our examples we used a single core of an Intel Core i7 CPU at 3.1 GHz. All the timings are summarized in Table 1. It is difficult to compare the performance of our new approach with that of existing methods because of the conceptual differences. For instance, in contrast to pure collision detection methods, our approach handles detection and response at the same time. Given the air mesh, (continuous) collision detection at the beginning of the time step is almost free since it only amounts to iterating through all air elements and testing for inverted volumes. In our case, most of the time is spent resolving collisions. One advantage of our method is that it constantly checks for new collisions during this process. Also, our method is targeted at a special type of problem so the performance differences highly depend on the given scenario.

For a proof of concept we first implemented a 2D version of our method. Figure 6 shows the most basic scene, a box in a rectangular domain. The scene contains 4 particles, 5 edge constraints for the rectangle and 8 air constraints. Simulating the particles with the given constraints yields the expected behaviour of the box colliding with the boundary. When the box rotates too much, the air elements lock. This problem is solved by turning on the air optimizer.

The same concept works for larger scenes too. Figure 2 shows 60 boxes and one rope all interacting properly by keeping the volumes of the air elements positive. To demonstrate the fact that the air mesh has a memory of the relative positions of objects, we set all y-values of the particles to zero. After releasing this constraint the objects recover both their shape and their order. This scene runs at more than 150 fps.

The third scene depicted in Figure 3 shows that the approach works for more complicated shapes as well. Again, after compressing the entire scene, the objects recover ad pop up on the correct side of the rope. Due to the simplicity of 2D mesh optimization, all the scenes take less than 5 ms per frame.

After verifying that the method works in 2D, we wrote a 3D solver as well. Figures 5 shows the equivalent of the multiple boxes scene in 3D. Again, we get the correct behaviour by keeping the volume of all air tetrahedra positive. After applying gravity, the boxes drop on the ground and stand stably without jittering. When applying random forces, the boxes move freely due to 3D mesh optimization. As expected, with 80 ms, mesh optimization takes significantly more time in 3D and is the bottle neck of this simulation. For this particular scenario, a traditional rigid body solver would be the better choice.

In the scene in Figure 8 of *The Town Musicians of Bremen* we performed mesh optimization only every fourth frame without introducing notable artifacts. The underlying simulation meshes are shown on the right of Figure 8.

Finally we tested our approach on two more realistic models, the sumo fighter shown in Figure 9 and the dancer from Figure 1. In

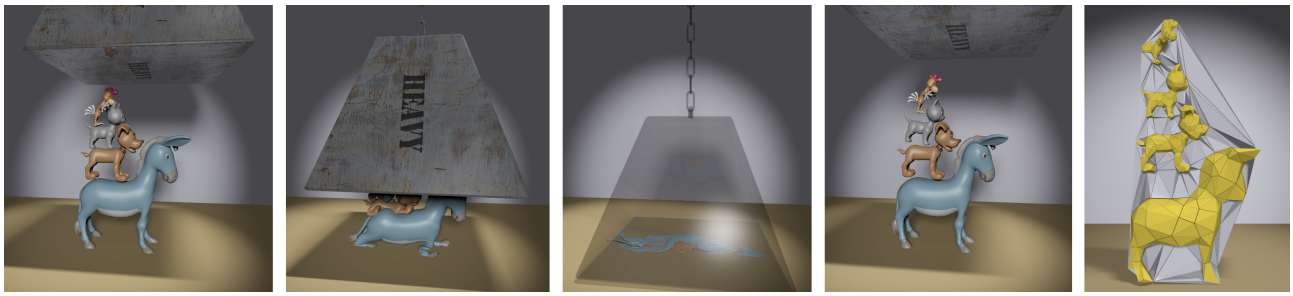

Figure 8: *Scene of the German fairy tale "Town Musicians of Bremen". The four animals are flattened completely. Besides the restoration of each character, the spatial order is conserved as well after lifting the weight. The underlying simulation mesh is shown on the very right with the object collision meshes in yellow and the air mesh in gray.*

Figure 9: *Resolving collisions with the local air meshes shown in white. The second and fourth images show the character with the air meshes turned off. The unresolved collisions are clearly visible at the armpits and between the thighs and the belly. Turning the air meshes back on resolves the collisions smoothly and stably.*

Scene	object elements	air elements	object sim	iters	air sim	iters	air optimization	total
2D boxes	120	416	3.3 ms	20	1.6 ms	20	1.6 ms	6.5 ms
2D objects	269	165	3.4 ms	20	0.5 ms	20	0.9 ms	4.8 ms
3D boxes	250	2775	3.5 ms	10	1.7 ms	10	80 ms	85.2 ms
3D characters	1919	2739	6.5 ms	10	2.1 ms	10	14 ms	22.6 ms
sumo fighter	32,000	1700	35 ms	1	15 ms	10	-	50 ms
dancer	47,000	165,000	119 ms	10	520 ms	20	-	639 ms

Table 1: *Sizes and timings of our example scenes. The first two columns denote the number of elements (triangles or tetrahedra) of the object and air meshes, respectively. The next four columns state the time taken by the simulation of the objects, the object iteration count, the time taken for simulating the air mesh and the air mesh iteration count. Finally we give the time for the air mesh optimization. In the last two scenes, we turned optimization off.*

Figure 10: *Our method is able to smoothly resolve complicated entangled states. Left: configuration after turning off the air mesh. Middle and left: The skirt finds its way smoothly into untangled state.*

Figure 11: *The air mesh used for the dancer scene of Figure 1.*

both cases, our approach yields plausible results even with mesh optimization turned off.

Handling collisions of the tissue under the armpits is a challenging problem. We solved this problem by simply adding small air meshes as shown on the left of Figure 9. The second and fourth images show the simulation with the air meshes turned off. Activating them resolves the collisions smoothly. There is no jittering due to varying collision sets as is some existing methods because the number and order of the constraints stays fixed. Two additional air meshes resolve the collisions between the thigh and the belly too. Because the air meshes are so small, the time for collision handling is almost negligible.

To simulate the dancer with the multi-layered skirt shown in Figure 1, we used an air mesh of 165k tetrahedra containing the 47k vertices of the cloth meshes and the vertices of the characters legs. Figure 11 shows the resulting mesh. The entire scene still runs at about two frames per second. In Figure 10 and the accompanying video, we show what happens when the iteration count is set to zero. Since collision with the character is also handled by the air mesh, the skirt just drops straight down through the legs and the layers swing across each other. After turning the air mesh back on, the skirt smoothly finds its way into a legal state.

6 Conclusion and Future Work

We have introduced the concept of air meshes. Preventing the air elements from inverting yields robust collision detection and collision response. An important feature of our approach is the fact that the air mesh stores information of the relative positions of objects which allows the recovery from entangled states. Therefore, the method is well suited for complex objects in close proximity such as colliding layers of tissue or multi-layered clothing.

We showed, that it is also applicable to typical rigid body scenes with freely moving objects. In 2D, the method is robust, fast and simple to implement and therefore, an attractive method for physical effects in mobile 2D games. In 3D, due to the relatively expensive mesh optimization operations, traditional collision detection methods are better suited for classical rigid body scenes. Fortunately, mesh optimization can be turned off in the scenarios we target without introducing disturbing visual artifacts.

There are various possibilities for future work. For instance, to implement vertex insertion proposed by Klingner and Shewchuk [2007] air particles could be introduced. An other idea would be to dynamically create and remove air elements as needed or to use air meshes as a broad phase of a hybrid method.

Parallelizing our method is straight forward. In fact, we already have a GPU accelerated framework for the simulation of various materials and objects based on Position Based Dynamics. All we have to do to integrate our algorithm is to add a new constraint type, namely a unilateral volume constraint. We are currently working on such an integration.

7 Acknowledgments

We would like to thank NVIDIA and the PhysX team for their support and the anonymous reviewers for their valuable feedback.

References

AGARWAL, P. K., BASCH, J., GUIBAS, L. J., HERSHBERGER, J., AND ZHANG, L. 2000. Deformable free space tilings for kinetic collision detection. In *International Journal of Robotics Research*, 83–96.

ALLARD, J., FAURE, F., COURTECUISSE, H., FALIPOU, F., DURIEZ, C., AND KRY, P. G. 2010. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph. 29*, 4 (July), 82:1–82:10.

BARAFF, D., WITKIN, A., AND KASS, M. 2003. Untangling cloth. *ACM Trans. Graph. 22*, 3 (July), 862–870.

BARBIČ, J., AND JAMES, D. L. 2010. Subspace self-collision culling. *ACM Trans. Graph. 29*, 4 (July), 81:1–81:9.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July), 594–603.

BROCHU, T., EDWARDS, E., AND BRIDSON, R. 2012. Efficient geometrically exact continuous collision detection. *ACM Trans. Graph. 31*, 4 (July), 96:1–96:7.

CIVIT-FLORES, O., AND SUSN, A. 2014. Robust treatment of degenerate elements in interactive corotational fem simulations. *Computer Graphics Forum 33*, 6, 298–309.

CURTIS, S., TAMSTORF, R., AND MANOCHA, D. 2008. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '08, 61–69.

FAURE, F., BARBIER, S., ALLARD, J., AND FALIPOU, F. 2008. Image-based collision detection and response between arbitrary volume objects. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, 155–162.

FREITAG, L. A., AND OLLIVIER-GOOCH, C. 1997. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering 40*, 21, 3979–4002.

GOVINDARAJU, N. K., REDON, S., LIN, M. C., AND MANOCHA, D. 2003. Cullide: Interactive collision detection between complex models in large environments using graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, HWWS '03, 25–32.

GOVINDARAJU, N. K., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M. C., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph. 24*, 3 (July), 991–999.

GRIMM, J., AND GRIMM, W. 1812. Town musicians of bremen. In *Grimm's Fairy Tales*, vol. KHM 27.

HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2004. Detection of collisions and self-collisions using image-space techniques. In *Proceedings of Computer Graphics, Visualization and Computer Vision*, 145–152.

HUTTER, M., AND FUHRMANN, A. 2007. Optimized continuous collision detection for deformable triangle meshes. *Journal of WSCG 15*, 1-3, 25–32.

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

IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph. 26*, 3.

KLINGNER, B. M., AND SHEWCHUK, J. R. 2007. Agressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, 3–23.

LEVIN, D. I. W., LITVEN, J., JONES, G. L., SUEDA, S., AND PAI, D. K. 2011. Eulerian solid simulation with contact. In *ACM SIGGRAPH 2011 Papers*, ACM, New York, NY, USA, SIGGRAPH '11, 36:1–36:10.

MACKLIN, M., MÜLLER, M., CHENTANEZ, N., AND KIM, T.-Y. 2014. Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July), 153:1–153:12.

MEZGER, J., KIMMERLE, S., AND ETZMUS, O. 2003. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG 11*, 1-3.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, GI '04, 239–246.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, SIGGRAPH '05, 471–478.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr.), 109–118.

PAGANO, S., AND ALART, P. 2008. Self-contact and fictitious domain using a difference convex approach. *International Journal for Numerical Methods in Engineering 75*, 1 (July), 29–42.

PREZ-URBIOLA, R. E., AND RUDOMIN, G. I. 1999. Multi-layer implicit garment models. In *Shape Modeling International Proceedings*, 66–71.

SCHVARTZMAN, S. C., GASCÓN, J., AND OTADUY, M. A. 2009. Bounded normal trees for reduced deformations of triangulated surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, 75–82.

SCHVARTZMAN, S. C., PÉREZ, L. G., AND OTADUY, M. A. 2010. Star-contours for efficient hierarchical self-collision detection. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) 29*, 3.

SELLE, A., SU, J., IRVING, G., AND FEDKIW, R. 2008. Robust high-resolution cloth using parallelism, history-based collisions and accurate friction. *IEEE Transactions on Visualization and Computer Graphics, (TVCG) 15*, 2, 339 – 350.

SHEWCHUK, J. R. 2002. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. *Manuscript*.

SI, H. 2015. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw. 41*, 2 (Feb.), 11:1–11:36.

SIFAKIS, E., MARINO, S., AND TERAN, J. 2008. Globally coupled collision handling using volume preserving impulses. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '08, 147–153.

STOMAKHIN, A., HOWES, R., SCHROEDER, C., AND TERAN, J. M. 2012. Energetically consistent invertible elasticity. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 25–32.

TANG, M., CURTIS, S., YOON, S.-E., AND MANOCHA, D. 2008. Interactive continuous collision detection between deformable models using connectivity-based culling. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, ACM, New York, NY, USA, 25–36.

TANG, M., MANOCHA, D., AND TONG, R. 2010. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '10, 7–13.

TANG, M., MANOCHA, D., YOON, S.-E., DU, P., HEO, J.-P., AND TONG, R.-F. 2011. Volccd: Fast continuous collision culling between deforming volume meshes. *ACM Trans. Graph. 30*, 5 (Oct.), 111:1–111:15.

TANG, M., TONG, R., WANG, Z., AND MANOCHA, D. 2014. Fast and exact continuous collision detection with bernstein sign classification. *ACM Trans. Graph. 33* (November), 186:1–186:8.

TESCHNER, M., HEIDELBERGER, B., MUELLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. 47–54.

TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., AND RAGHUPATHI, R. 2005. Collision detection for deformable objects. *Computer Graphics Forum 24*, 1, 61–81.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2006. Resolving surface collisions through intersection contour minimization. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, 1154–1159.

WANG, B., FAURE, F., AND PAI, D. K. 2012. Adaptive image-based intersection volume. *ACM Trans. Graph. 31*, 4 (July), 97:1–97:9.

WANG, H. 2014. Defending continuous collision detection against errors. *ACM Trans. Graph. 33*, 4 (July), 122:1–122:10.

WICKE, M., LANKER, H., AND GROSS, M. 2006. Untangling cloth with boundaries. In *Proceedings of VMV*, 349–356.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29*, 4 (July), 49:1–49:11.

WONG, S.-K., AND CHENG, Y.-C. 2014. Gpu-based radial view-based culling for continuous self-collision detection of deformable surfaces. *The Visual Computer*, 1–15.

WONG, W. S.-K., BACIU, G., AND HU, J. 2004. Multi-layered deformable surfaces for virtual clothing. In *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, New York, NY, USA, VRST '04, 24–31.

WONG, S.-K., LIU, C.-M., BACIU, G., AND YEH, C.-C. 2010. Robust continuous collision detection for deformable objects. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, ACM, New York, NY, USA, VRST '10, 55–62.

WONG, S.-K., LIN, W.-C., HUNG, C.-H., HUANG, Y.-J., AND LII, S.-Y. 2013. Radial view based culling for continuous self-collision detection of skeletal models. *ACM Trans. Graph. 32*, 4 (July), 114:1–114:10.

WONG, S.-K., LIN, W.-C., WANG, Y.-S., HUNG, C.-H., AND HUANG, Y.-J. 2014. Dynamic radial view based culling for continuous self-collision detection. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, 39–46.

YOON, S.-E., KIM, Y. J., HARADA, T., KIM, Y. J., AND YOON, S.-E. 2010. Recent advances in real-time collision and proximity computations for games and simulations. In *ACM SIGGRAPH ASIA 2010 Courses*, ACM, New York, NY, USA, SA '10, 22:1–22:110.

ZHANG, X., AND KIM, Y. 2012. Simple culling methods for continuous collision detection of deforming triangles. *IEEE Transactions on Visualization and Computer Graphics 18*, 7, 1146–1155.

ZHENG, C., AND JAMES, D. L. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph. 31*, 4 (July), 98:1–98:12.