Open access • Proceedings Article • DOI:10.1145/2821592.2821615

GPU ray-traced collision detection for cloth simulation — Source link

François Lehericey, Valérie Gouranton, Bruno Arnaldi

Institutions: French Institute for Research in Computer Science and Automation

Published on: 13 Nov 2015 - Virtual Reality Software and Technology

Topics: Collision detection, Physics engine, General-purpose computing on graphics processing units and Ray tracing (graphics)

GPU Ray-Traced Collision Detection for Cloth Simulation

François Lehericey, Valérie Gouranton, Bruno Arnaldi

GPU Ray-Traced Collision Detection for Cloth Simulation

François Lehericey* Valérie Gouranton† Bruno Arnaldi‡
INSA de Rennes, IRISA, Inria
Campus de Beaulieu, 35042 Rennes cedex, France

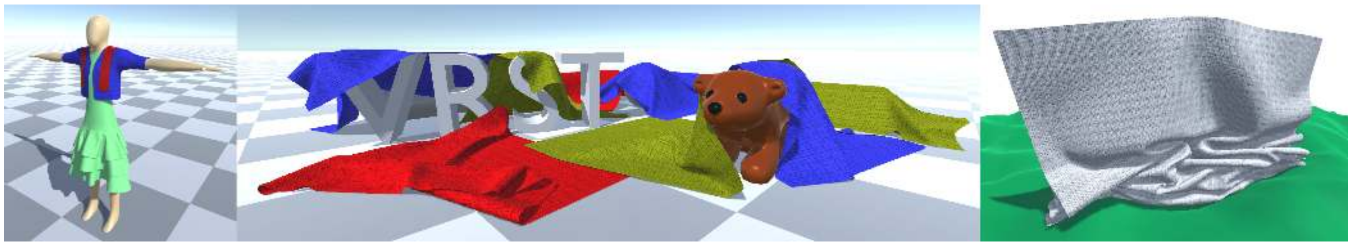

Figure 1: *Our method can perform collision detection between clothes and handle self collision detection. With our method, the layer of a three-layered dress can be simulated as well as the wrinkle formation of a sheet falling on the ground.*

Abstract

We propose a method to perform collision detection with cloths with ray-tracing. Our method is able to perform collision detection between cloths and volumetric objects (rigid or deformable) as well as collision detection between cloths (including auto-collision). Our method casts rays between objects to perform collision detection, and an inversion-handling algorithm is introduced to correct errors introduced by discrete simulations. GPU computing is used to improve the performances. Our implementation handles scenes containing deformable objects at an interactive frame-rate, with collision detection lasting a few milliseconds.

CR Categories: Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Raytracing

Keywords: collision detection, narrow-phase, ray-tracing

1 Introduction

Collision detection is an essential task for physics simulation; it is responsible for detecting colliding objects and producing contact points needed to compute an accurate physics response. Collision detection is a bottleneck in virtual reality applications due to the complexity of the environments that we try to simulate (e.g. large environments with complex objects such as deformable objects).

Nowadays, collision detection is decomposed in two phases: broad-phase and narrow-phase. The broad-phase takes as input the whole set of objects present in the simulation, and output a list of pairs of objects that might be in collision with simple tests (with bounding volumes). The narrow-phase takes these pairs and performs more accurate tests to detect collision and output contact points for the physics response. Our method belongs to the narrow-phase category and, thus, performs collision detection on pairs of objects.

We propose a narrow-phase collision detection algorithm for cloths that use ray-tracing to detect collisions between two objects. Our method is able to perform discrete collision detection between a piece of cloth and a volumetric object (including non-convex objects) as well as collision detection between cloths (including auto-collision). This algorithm is paired with a method to correct collision artefacts between cloths caused by discrete simulations. With discrete simulations, artefacts can happen when objects have movement with higher amplitude than their sizes in a single time-step. This problem is critical with cloths because of their small thickness.

2 Related Work

We review related work that is the most relevant to our contributions. For an exhaustive review on collision detection we refer the reader to [Teschner et al. 2005] and [Kockara et al. 2007].

Among all the narrow-phase methods, image-based methods use rendering techniques to detect collisions and generally do not need pre-processing as they use the objects in the form they used in rendering. This makes image-based methods highly suitable for deformable objects. These methods are often adapted for GPU implementations as GPUs are designed to execute rendering algorithms. Among rendering techniques, ray-tracing is commonly used to perform collision detection. [Wang et al. 2012] use ray-tracing to compute a layered depth image with a higher density around small features with a non-uniform ray sampling. [Hermann et al. 2008] detect collision by casting rays from the vertices of the objects. Rays are cast in the inward direction and the ones that hit the interior of another object detect collisions. [Lehericey et al. 2013b] cast secondary rays in the outward direction to detect prediction (possible future collisions), which can be used to improve the detection and the physics response.

Collision detection for cloth simulation is a complex challenge and has been extensively studied. All pairs of features (vertex, triangle, ...) between objects have to be tested for collision; therefore, a query for a single pair has a complexity of $O(n \times m)$, with n and m being the number of features to test on each object. This high complexity is exacerbated by the constant deformations to which

*francois.lehericey@irisa.fr
†valerie.gouranton@irisa.fr
‡bruno.arnaldi@irisa.fr

an object can be subject, which makes the design of accelerative methods complex. Nevertheless, numerous accelerative methods have been proposed to improve the performances by culling unnecessary tests. [Teschner et al. 2003] use spatial hashing to improve the performances. [Tang et al. 2009] perform continuous collision detection with hierarchical culling and take into account triangle connectivity.[Zheng and James 2012] perform self-collision culling by computing the amount of energy required by a deforming object in order to enter into self-collision.

Some methods consider the space between the objects. [Müller et al. 2015] tessellate the air between the objects. Inversions are handled by forcing the volume of tessellated elements to be positive. If large movements are present in the simulation, the tessellation needs to be optimised. Treating inverted elements is linked to finite element simulations [Stomakhin et al. 2012].

3 Ray-Traced Collision Detection for Cloths

With volumetric objects (rigid or deformable), collision detection can be performed by casting rays from the vertices inside the object, as outlined in [Hermann et al. 2008], and predictive rays outside the object, as outlined in [Lehericey et al. 2013b]. In these methods, volumetric objects are represented by their surface.

We propose to extend this narrow-phase method to handle cloths. In our method, cloths are represented as a surface, but instead of representing the limits of the object, the surface represents the centre layer of the object. Cloth interior expands towards both sides of the supporting surface up to a distance e (e = half cloth thickness), with e small regarding the dimensions of the cloth. Figure 2 shows our cloth representation compared to volumetric objects.

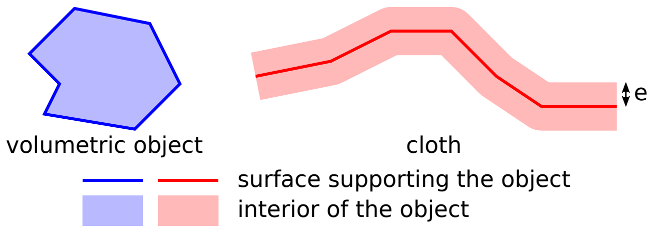

Figure 2: *Representation of volumetric objects and cloth.*

Our method casts rays from each vertex of each object. Each pair $(A; B)$ is decomposed in two tests: $(A \rightarrow B)$ and $(B \rightarrow A)$, where rays are cast respectively from A to B and B to A. Each test $(A \rightarrow B)$ performs a collision query of each vertex of A against B. The combination of the two tests enables us to test all of the vertices from both objects. Performing only one of the two tests could lead to false-negative detection. We decompose the collision query of a pair $(A; B)$ in two separate tests to take into account each object's specificity. A and B can have different properties; each of these objects can be a volumetric object or a cloth. Given the differences in the representation of objects, all of the cases need to be specialised.

Auto-collision detection for cloths is handled in the same way as collision between two different cloths. To handle auto-collision detection we simply add for each cloth C a test $(C \rightarrow C)$.

Our method outputs contact points that will be used by the physics response. Contact points give a pair of features – one from each object that is in an interpenetration state. In our method these features are a vertex (from which a ray was cast) and a triangle (which was hit by the ray). Each feature is coupled with a normal; this normal gives the optimal direction to push the objects to separate them.

Our method works in three steps (detailed in the next sections):

- We cast rays from the vertices of the objects. These rays are responsible for the detection of colliding features and the detection of possible future colliding features (predictions).
- For the cloth vs. cloth test we apply an inversion detection algorithm. This test detects collision artefacts introduced when movements are too important in a single time-step.
- We generate contact points for the physics response. Contact points are generated from the result of the ray-tracing (and potentially corrected with inversion detection). Some post-processing is applied to ensure a correct physics response.

3.1 Ray-Traced Detection

The first step in performing collision detection on a pair of objects is to cast rays from each vertex of each object of the pair. Two rays are cast from each vertex; one in the direction of the normal and one in the opposite direction of the normal.

One problem to address is to avoid detecting conflicting contact points. Contact points must be inside the colliding objects and the contact normal should give the direction to the shortest path to separate the objects. Conflicting contact points are unreliable contact points detected either outside the objects or with a normal which does not give a correct path to separate the objects. These conflicting contact points will cause errors in the physics response by generating opposing forces that can lock objects together.

To avoid these conflicting contact points, Hermann et al. proposed using two conditions when testing collisions between two volumetric objects: (1) the ray must hit the target object before leaving the source object, and (2) the ray must hit the inside of the target object. These two conditions ensure that there will be no conflicting contact point that would lock the objects together in the physics response. An example can be found in Figure 3, where the crossed-out arrows are eliminated because it does not satisfy condition (1) or (2).

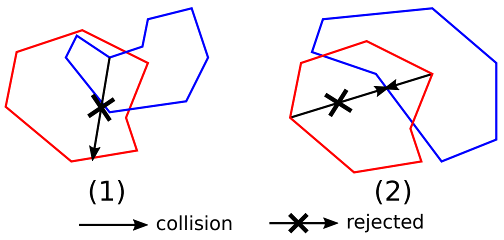

Figure 3: *Illustration of Hermann et al. conditions.*

Our new contributions include three new elementary tests that allow the introduction of cloths into our collision detection method: volumetric → cloth, cloth → volumetric, and cloth → cloth.

volumetric → cloth

For a volumetric object the ray cast in the direction of the normal goes outside the object (outward ray) and the ray cast in the opposite direction of the normal goes inside the object (inward ray). Figure 4.a gives an example of rays classified as collision, prediction or rejected by the inward ray or the outward ray.

For the inward ray, the second condition proposed by Hermann et al. to avoid conflicting contact is not feasible. Collision tests are processed on the supporting surface of the cloth (the centre layer), which makes impossible to know whether the ray hits inside or outside the cloth. To avoid conflicting contact we propose to use a modified version of condition (1). Collision is detected if the length

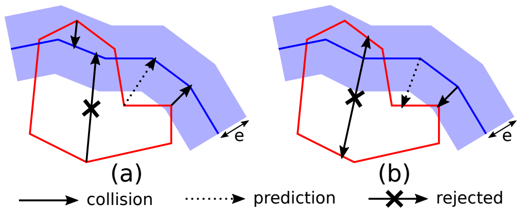

Figure 4: *Example of an accepted, predictive and rejected rays in (a) a volumetric $\rightarrow$ cloth test and (b) a cloth $\rightarrow$ volumetric test.*

of the ray does not exceed half the exit distance (distance of the ray when it exits the source object). If the ray has a length higher than half the exit distance, then the cloth is closer to the other side of the object; collision will be detected on this other side.

In [Lehericey et al. 2013b] method, the outward ray is only used for collision predictions. With cloths we have to take into account the thickness of the cloth. Our solution is to classify cloth features detected by the outward ray as collision if the length of the ray is inferior to e; otherwise we classify it as a prediction.

cloth $\rightarrow$ volumetric

With cloths, both rays are cast inside the object and exit it after traversing a distance e. Unlike volumetric objects, both rays cast from a cloth are treated in the same way. Figure 4.b gives an example of collision detected from a cloth towards a volumetric object.

When a ray cast by a cloth hits a volumetric object we have two cases: (I) the ray hits the inside of the volumetric object, or (II) the ray hits the outside of the volumetric object. In case (I), both rays cast from the same vertex will hit inside the other object (because the cloth vertex is inside the other object). We cannot use both rays to detect collision, as it would give an unreliable response (conflicting contact). In that case we classify the ray with the shortest length as a collision and we discard the other. In case (II), we detect a collision if the ray has a length inferior to e (to take into account the cloth thickness); otherwise we classify it as a prediction.

cloth $\rightarrow$ cloth

Like the cloth $\rightarrow$ volumetric case, both rays are cast inside the object and exit it after traversing a distance e. When detecting collision between two clothes, we have to take into account both cloth thicknesses (illustrated in Figure 5). When a ray cast by a cloth hits another cloth we detect a collision if the ray has a length inferior to $e+e'$ (with e and e' the half thickness of each cloth); otherwise we classify it as a prediction.

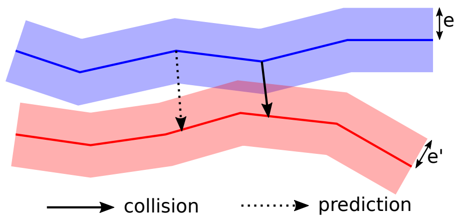

Figure 5: *Example of an accepted and predictive ray in a cloth $\rightarrow$ cloth test. The ray is accepted if its the length in inferior to* $e + e'$.

3.2 Inversion Handling

With discrete physics simulations, collision detection is executed after objects move with a delta-time. In this case, collision detection is performed when objects are in interpenetration and the correct response to apply has to be found. When small objects are present, they can traverse each other during a single time-step. This problem is highly sensitive with cloths because of their small thickness and the relative independent movement of the vertices.

When inversion happens between two cloths (or in self-collision), the two cloths behave like they are glued together. This is explained by the conflicting constraints between the cloths. The physics response tries to maintain each vertex on the side on which it was detected. Inversions also provoke visual artefacts. Inverted vertices are not rendered on the correct side of the other cloth and cause visual errors (e.g. Figure 6).

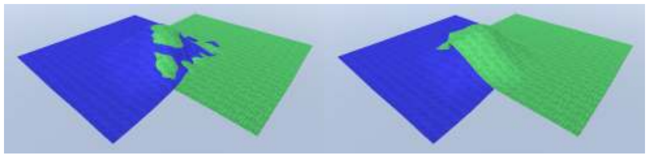

Figure 6: *Two pieces of cloth fall on a ball. Without inversion detection (left image), both cloths become entangled. With inversion detection (right image), no error are present in the simulation.*

Our solution is to check the side of the colliding vertex regarding the colliding triangle and compare it with the side of the same vertex regarding the same triangle before time integration. If the vertex changes side regarding the triangle, we consider that we have an inversion; in such cases, we invert the direction of the normals of the contact points. The inversion of the direction of the normals will cause the physics response to push the vertex and the triangle in the opposing direction thus untangling the cloths.

When an inversion is detected, if the contact is classified as a prediction, then we need to reclassify it as a collision. This can happen when the vertex moves a distance higher than the thickness of the cloths in a single time-step. If the contact point is not reclassified, then it will be ignored by the physics response, which would leave the cloths in a tangled state. Should the same contact be detected in the next steps, the inversion detection algorithm would not be able to correct it as it only works on the last step result.

3.3 Contact Points Generation

After collision is detected and inversions are corrected we need to generate contact points for the physics response. To improve the reliability we use ray re-projection [Lehericey et al. 2013b] to comply with the MTD. The MTD (minimum translational distance [Cameron and Culley 1986]) is the shortest relative translation that puts the two objects in contact (and not in interpenetration). Ray re-projection guarantees that the contact point gives the shortest translation to separate the object relatively to the detected triangle by projecting the ray on the normal of the detected triangle.

The projection of the ray modifies the length of the ray and makes it shorter. With detection depending on the length of the ray, ray re-projection needs to be applied before performing the classification of contact points to avoid misclassifying an actual collision as a prediction. These errors, when present in a simulation, cause instabilities. Contact points are not detected at the distance that the physics response puts them, resulting in an alternation between a detected and undetected state.

Contact points are detected on the surface supporting the object. For cloths this surface does not represent an external surface of the cloth. To put the contact point of a cloth on the exterior surface of the object, the contact point should be translated in the opposite direction of the contact normal by a distance e.

4 Results

We tested our ray-traced collision detection algorithm with a position-based physics response [Bender et al. 2013]. The simulation is implemented for GPU with OpenCL and is executed on an AMD FirePro W9100 at 60 frame per second.

Our method can use any existing ray-tracing algorithm to perform collision detection; in our tests we used three ray-tracing algorithms. For rays cast on rigid objects we use a stack-less BVH traversal. This algorithm uses a bounding volume hierarchy (BVH) as an accelerative structure [Wald et al. 2007], with a stack-less ray traversal designed to maximise performance on GPU. For rays cast on deformable objects we use a basic ray-tracing algorithm. This algorithm does not use any accelerative structure that would need to be updated. For rays cast on both rigid and deformable objects, we use an iterative ray-tracing algorithm in addition [Lehericey et al. 2013a]. This algorithm can only be used when small displacements are present in the simulation over time. This algorithm can be used on both rigid and deformable objects, as the accelerative structure does not need to be updated when deformation occurs. The core idea of this method is to update the previous time-step when small displacements occur between objects; otherwise standard ray-tracing algorithms are used.

Figure 1 show the experimental scenes we used for test. In the first scene, a static mannequin (composed of 9800 vertices) wears a dress (11,000 vertices), a jacket (2700 vertices) and a scarf (3800 vertices). In this scene, collision detection takes an average of 4.3 milliseconds per time-step. In this scene our method is able to handle a complex scenario where three layers of clothes are stacked. Up to 16,000 contact points are detected in every step between clothes and with the mannequin. The scarf lies on the jacket and touches the dress, the jacket lies on both the dress and mannequin, and no interpenetrations are present in the simulation. Auto-collisions are accurately detected on the three-layered dress, resulting in a visible volume and wrinkles.

In the second scene, eight sheets (each composed of 4200 vertices) are dropped on a static scene (composed of 9000 vertices). Collision detection takes an average of 5.3 milliseconds per time-step.

The third scene shows a stress test for self-collision detection. A square piece of cloth is dropped on its side on an irregular ground. Our method is able to detect auto-collision occurring while hitting the ground resulting in the formation or wrinkle without self-interpenetration. Up to 3800 contact points are detected per time-step, including collisions between the cloth and the ground and auto-collisions on the cloth.

5 Conclusion

We presented a method to perform collision detection for cloths using ray-tracing. Our method is able to perform collision detection between volumetric objects (rigid or deformable) and cloths as well as collision detection between cloths (including auto-collision). Our inversion-handling method is able to rectify errors introduced by discrete simulation. Our implementation showed that our method can be used in real-time simulations of deformable objects and is able to take advantage of GPU computational power.

As in any discrete simulations, the velocities of the objects need to be limited depending on the frame-rate of the simulation. If objects move of a distance higher than their sizes in a single time-step, they can pass through another object without any collision being detected. Adjusting the direction of each ray toward the velocity vector of each vertex relatively to the movement of the target would give an approximation to continuous collision detection.

In future work we want to integrate untangling algorithms that work in the long term (several time-steps) to have more robust simulations. In addition, we want to investigate temporal consistency between cloths to improve performances. Estimation of the distance to collision (which is possible with collision predictions) paired with a measurement of displacement could be used to perform culling on vertices until a collision is possible.

References

BENDER, J., MÜLLER, M., OTADUY, M. A., AND TESCHNER, M. 2013. Position-based methods for the simulation of solid objects in computer graphics. *EUROGRAPHICS 2013 star*.

CAMERON, S., AND CULLEY, R. 1986. Determining the minimum translational distance between two convex polyhedra. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, IEEE, 591–596.

HERMANN, E., FAURE, F., AND RAFFIN, B. 2008. Ray-traced collision detection for deformable bodies. In *GRAPP 2008*, INSTICC, 293–299.

KOCKARA, S., HALIC, T., IQBAL, K., BAYRAK, C., AND ROWE, R. 2007. Collision detection: A survey. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, IEEE.

LEHERICEY, F., GOURANTON, V., AND ARNALDI, B. 2013. New iterative ray-traced collision detection algorithm for gpu architectures. In *Proceedings of the 19th ACM VRST*, 215–218.

LEHERICEY, F., GOURANTON, V., ARNALDI, B., ET AL. 2013. Ray-traced collision detection: Interpenetration control and multi-gpu performance. *JVRC*, 1–8.

MÜLLER, M., CHENTANEZ, N., KIM, T.-Y., AND MACKLIN, M. 2015. Air meshes for robust collision handling. *ACM Trans. Graph. 34*, 4 (July), 133:1–133:9.

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, 25–32.

TANG, M., CURTIS, S., YOON, S.-E., AND MANOCHA, D. 2009. Iccd: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics 15*, 544–557.

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANTES, D., AND GROSS, M. H. 2003. Optimized spatial hashing for collision detection of deformable objects. In *VMV*, vol. 3.

TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., ET AL. 2005. Collision detection for deformable objects. In *Computer graphics forum*, vol. 24, Wiley Online Library.

WALD, I., BOULOS, S., AND SHIRLEY, P. 2007. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics (TOG) 26*, 1, 6.

WANG, B., FAURE, F., AND PAI, D. K. 2012. Adaptive image-based intersection volume. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4.

ZHENG, C., AND JAMES, D. L. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM TOG 31*, 4, 98.