Collision Detection in Cloth Simulation

Tobias Schleser*
Student of Computer Science
Vienna University of Technology, Spring 2007

Abstract

The simulation of cloth is a demanding task that has become important throughout the past years. The importance of interactive environments for deformable objects like cloth or fabrics in general but also human skin has been increased as the entertainment industry is aiming for constantly improved realism. Also the cloth industry is starting to make product catalogues be available online in interactive ways. The only possibility to grant maximum freedom to the user (who is the customer) while exploring the fabric is to model products virtually. This imposes new challenges to preserve realistic and equally important accurate "look and feel". The mentioned domains have in common that collision detection is needed during the rendering phase. Fabrics might look perfectly realistic by utilizing modern texture functionalities but if the geometric modeling of collisions is messed up, i.e. the cloth intersects itself or the model it is fitted to, the impression is quickly destroyed. Collision detection of deformable objects is a demanding task, however, and interactivity is not easy to reach. In this paper I present some traditional and some new approaches to overcome the problem of complex deformable objects interacting with themselves or a virtual world.

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

Keywords: cloth modeling, collision detection, knitwear, deformable surfaces

1 Introduction

The modeling and simulation of knitwear and other fabrics has been an area of active research over the past fifteen years. The computer graphics community is aiming for constant enhancement of modeling and rendering techniques in order to produce virtual fabrics with a realistic "look and feel" for the cloth. At the same time, deformable surfaces induce a number of difficulties. One of these is the problem of collision detection for non rigid objects that will be discussed thoroughly in this paper.

Besides that, there are a number of issues one has to deal with when it comes to realistic cloth simulation. In the modeling process, the decision about the mesh structure has to be made. The most common approach is the use of triangle meshes but collision detection has been carried out on time-dependent parametric surfaces [Herzen et al. 1990] and subdivision surfaces [Grinspun et al. 2001] as well. Another representation would be particle systems as used in [Fuhrmann et al. 2003]. Some of the approaches introduced here work with arbitrary mesh structures and some require pretty strict restrictions (i.e. closed meshes, mesh topology, etc.).

Also, the cloth model itself, namely the internal cloth dynamics, need to be constructed. A simple but common model that is used by [Bridson et al. 2002] for example (the approach is discussed in more detail later) is the following mass-spring model (details in [Louchet et al. 1995]): springs are used to connect the immediately neighbouring particles (masses) that are arranged in a rectangular grid. Additionally, there are diagonal springs that support shearing as well as springs connected to every second node to avoid bending. The model is adapted by different authors, i.e. Bridson et al. made corners and edges heavier to produce more realistic results. Again, many collision detection algorithms are indepedent of the underlying model.

Besides these structural issues, also the rendering process itself is an other crucial issue for realistic virtual cloth. The topic is presented well in [Sattler et al. 2003] where a *bidirectional texture function (BTF)* is calculated from measurements of real-world fabrics to obtain valuable material for creating a mesostructure, the fine but visible-to-the-eye structure of a given material.

The issues mentioned above are of great importance for realistic cloth rendering but as a basis have the need for efficient collision detection in common. Even if the rendered fabric is of high quality and looking realistic: if the collision detection fails and the fabric intersects itself or backfaces can be seen, the impression is destroyed.

There are issues, though, why collision detection is hard to perform for deformable, time dependent surfaces. The most obvious is the large number of possible collisions: each point on the surface might possibly collide with almost every other point. This fact gives rise to performance problems. Naive methods that try to monitor all possible collision points quickly halt the simulation. To overcome this, some approaches do not try to prevent 100 per cent of the collisions but prevent the most likely ones (a considerably less amount) and deal with the collisions that occured in a post processing step. The above mentioned method of [Bridson et al. 2002] dealt with the large number of potentially colliding points this way. Their approach will be discussed in more detail later.

Also, a natural attribute of fabrics is their thinness. It is obvious that this can lead to disturbing artifacts even if only a rather small amount of collisions is not detected due to the fact that backfaces are likely to appear in this case.

A lot of work has been performed on the collision detection topic during the past two decades. It is not possible to cover it all here so I will concentrate on some interesting approaches and describe them in more detail. The rest of this paper is organized as follows: In Section 2 a number of collision detection methods will be presented. Among them bounding object hierarchies or distance field approaches. Also, an image based method is presented that is fundamentally different than the other approaches. While most traditional methods handle collision in object space, the image based method operates in image space making use of graphics cards depth and stencil buffers. The section covers experimental results, too. I give a summary and conclude in Section 3.

2 Collision detection of Fabrics

As mentioned above collision detection for deformable surfaces is a complex domain and the search for efficient methods is subject to active research. Rigid body collision detection on the other hand

*e-mail: e0225349@student.tuwien.ac.at

has been performed for decades (i.e. ever since virtual worlds in 2D or three dimensional space have been build in computer graphics). Although it is still an active area of research, the community gets more and more attracted by the more difficult problem of collisions in the domain of fabrics and other deformable surfaces.

Collision detection for cloth and other fabrics is challangeing due to new issues, among them:

- Fabrics are time dependent deformable surfaces. Therefor, not only the collision detection but also the mesh structure needs to be recalculated for every time step. Together, this is computational problematic.
- There is a large number of possible collisions as in deformable meshes potentially every node can collide with any triangle other than the triangles it is part of and every edge can collide with any other edge except for its immediate neighbor edges.
- The thinness of fabrics (and the fact that usually backsides are not modelled) makes missed collisions problematic: even few can cause visually unpleasing results not only of fabrics intersecting themselves but also by visible backfaces.

In the following sections I will present some interesting approaches. Not all of them are new or designed especially for cloth rendering but might be adopted by few changes or expansions. I will first present two *bounding volume* approaches, afterwards I describe *distance field* methods. The third presented method deals with *voxelisation* to prevent collisions and is followed by a conceptionally different approach, namely an *image based* method (in contrast to the before mentioned object-space methods).

2.1 Hierarchies of bounding objects

[Bridson et al. 2002] focused their work on realistic collisions and friction (i.e. folds and wrinkles) in cloth animation. The work has been inspired by [Moore and Wilhelms 1988] who proposed repulsion forces and exact impulse-based treatment for contact and high velocity impact respectively. Moore and Wilhelms' paper can be considered as pioneer or early work in the domain of collision detection for deformable objects, but introduced basic concepts like the division of the problem into detection and response to collisions. Also they describe the commonly used spring model to react to collisions accurately.

In their paper, Moore and Wilhelms also discribe the today well known equations for testing, if a point intersected a triangle during a time step. The equation is presented as

$$P + (P' - P)t = P_0 + (P_1 - P0)u + (P_2 - P0)v$$

where P and P' represent the points' location at $t = 0$ and $t = 1$ (i.e. at the start and end of the time step), respectively. The P_i are triangle vertices and the parameters $u \geq 0$, $v \geq 0$ must suffice $u + v \leq 1$ in order to represent a triangles' parametrisation. If the equation can be solved (by matrix inversion) yielding $0 \leq t \leq 1$ and $u \geq 0$, $v \geq 0$ width $u + v \leq 1$, the point has collided with the triangle during the time step. Further on, the authors describe the equation also for moving triangles which is much more complex and yields a 5th order polynom that has to be solved (by binary search, as 5th order polynoms can not be solved analytically).

The second main contribution is the presentation of a collision response model using springs. A spring is temporarily attached to two points that are approximating one another or that have already interpenetrated. Some spring equation K/d is applied where K is the spring constant and d the distance. Thus, as d converges 0, the function goes to infinity. The determined force is applied in the opposite direction of the two approaching objects, thus pushing them appart. As extension, a second force (in the opposite direction) might be used to pay attention to the now receding objects. This way, they are not pushed appart too far. The extension models inelastic collision while without it an elastic collision is constructed.

[Bridson et al. 2002] relied on the basic cloth model using springs (that has been described in the introduction), too. Not to obtain visually unpleasing compressed or stretched triangles, a correction algorithm is used: Different from traditional approaches where a postprocessing step changes the position of nodes that have moved over 10 percent, Bridson at al. applied corrective impulses in cases where a velocity based prediction led to the result that the spring would be deformed more than 10 percent of its rest length. Doing so has two advantages.

- First, the behavior of real fabric is similar: until a certain percentage, deformation is relatively easy (i.e. little resistance), but above that, the critical deformation is reached and the fabric becomes stiff.
- Second, in contrast to the postprocessing approach, the nodes are not moved after the time step. This ensures that no new collisions (which are likely in the other case) are introduced.

The algorithm for friction and collision detection is as follows: For each time step, first the new positions and velocities as well as the average velocity is calculated with the cloth internal dynamics (notice the separation of collision detection and dynamics engine). Then, the initial positions are checked for proximity. This is done by a hierarchical bounding box algorithm. The hierarchy is build up at the beginning of the simulation with bounding boxes covering single triangles being the leaves and a box around the whole fabric being the root node (the inner levels are created by merging adjacent pairs of lower level boxes). Also, a bounding box hierarchy is created for leaf nodes that cover a triangle at its current position and at the predicted position after the time step to take the moving mesh into account. Now collision detection is performed for the hierarchies by testing the current box against the root node and, if intersection occured, recursively against its children. The test then breaks down to testing potentially colliding points against triangles and edges against edges.

Consequently, repulsion and friction are applied now. Repulsion is used to eliminate all or most of the collisions and thereby interpenetration while at the same time ensures a realistic behavior (the over all result is a mesh that seems to collide). When two points of the mesh have a close proximity in the order of the fabric thickness, repulsion forces that are oriented in the reversed moving direction of the points are applied. Since the impulse is inelastic (i.e. each point is moved away fom the other one at the average speed of their initial approximation speed), this way the cloth pieces are prevented from colliding. When pieces of the fabric are too close together, a second repulsion force is applied to model the compression of cloth fibers. These second repulsion forces are modeled as spring forces to obtain a more realistic behavior. Friction is done by Coulomb's friction model, see [Bridson et al. 2002] for details.

The authors emphasize that not all interpenetrations can be suppressed by their repulsion force approach. This is due to the fact that the positions of vertices on the mesh are checked at discrete time steps. Collisions, however, might occur also between the steps. Also, collisions might be oriented in the false direction. That is, points of colliding surfaces are oriented in a way, that the collision response would push the points together instead of appart. One way to deal with the fact is to rewind the simulation and process the occured collisions in chronological order. This is calculation ex-

pensive. However, a parallel approach that has been introduced by [Volino et al. 1995] was used. The method does not deliver a physically correct answer but yields a plausible solution. It has been introduced encouraged by the assumption that collisions mainly occur over larger areas. Therefore, neighboring points are used and a statistical method is applied to (a) decide if a collision occured and (b) determine the collision orientation. Once the orientation is computet, all affected vertices in the neighborhood are forced to have the majority's choice for orientation.

As a last step, post-processing using subdivision is performed in [Bridson et al. 2002]. This is done to get a smoother surface. Every edge is subdivided by a node that is put at the midpoint of the edge. Since the original mesh was free from interpenetrations, the subdivided mesh is too. Now a loop algorithm (see references in original paper) is used to find smoother positions what can introduce new collisions. These are handeled with the same algorithm that is used for the "real deformations" which arise from object movement or interaction (see above). Fortunately, this method converges fast (i.e. quickly no new necessary adjustments are found).

In contrast to the approach described above, the method of [James and Pai 2004] uses bounding spheres to model a hierarchy against which collisions can be tested. The hierarchy is labeled as *Bounded Deformation Tree (BD-Tree)* and the method basically consists of initially building the tree and updating it afterwards. The typical structure of hierarchical bounding volumes is layered, i.e. a parent node covers its child nodes completely. James and Pai, however, decided to build a wrapped tree. This technique does not require parent nodes to cover their complete children but to cover the mesh nodes these children represent. Figure 1 illustrates that this way, the bounding squeres can be kept smaller.

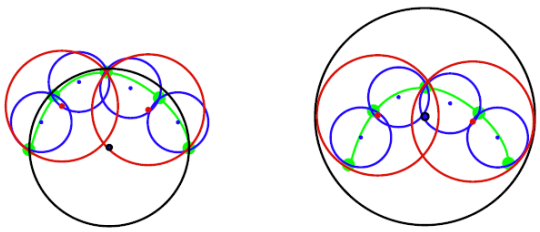

Figure 1: (right) traditional layered hierarchy with parent nodes covering complete child nodes; (left) the wrapped hierarchy that produces much smaller volumes; parent nodes cover the mesh nodes (green) of their children but not necessarily the whole child volumes; taken from [James and Pai 2004]

To build the decribed wrapped tree, first a traditional binary bounding sphere tree is built. Afterwards, all radii are made as small as possible while the centers of the spheres are left unchanged. This results in tighter bounding volumes and consequently more efficient calculations. Figure 2 shows the deformable model of a plastic chair that is bounded by a layered hierarchy of spheres in the top row and by a wrapped hierarchy on the bottom row (six levels of the hierarchy are shown). The tighter bounding at the final level 0 in the latter case is obvious.

When the object deforms, the spheres need to be updated. This can be done in a cost effective way: the center c is updated to c' by a function that takes the weighted positions of the contained points into account. The new radius r' is constructed using the triangle inequality and defined as $max_{i\in\Lambda}\left\|p'_i - c'\right\|$ where p'_i denotes the new positions of the node's points. For details see [James and Pai 2004]. Important is, that the spheres can be updated independently of each other and that resulting spheres are minimal again (thus, producing

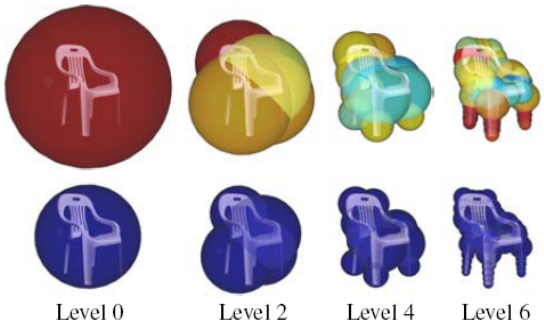

Figure 2: (top) a deformable plastic chair covered by a traditional layered bounding sphere hierarchy; (bottom) the tighter bounding of the same object by utilizing a wrapped hierarchy where coverage of the whole child nodes is not required for parent nodes; taken from [James and Pai 2004]

a new wrapped tree). Also, the update costs are independent of the numbers of object points covered in the bounding sphere.

The collision detection itself is performed by hierarchical testing of bounding spheres against each other. First, the topmost nodes of two objects are tested for intersection and if it occurs, the topmost node of one object is tested against the next level of the other one. Iteratively repeating this procedure (and switching the soles afterwards) finally yields the smallest bounding spheres that intersect each other. Here, traditional triangle-point intersection tests need to be performed (as explained at the beginning of this section).

There are a number of other hierarchical bounding volume methods that have been covered well in the community. Although they have not been optimized for deformable objects, two interesting approaches are mentioned (see references for further detail):

- [Zachmann 1998] introduces dynamically aligned *Discrete Oriented Polytop (DOP) Trees*. DOPs are convex polytopes and there are constraints for their orientation (i.e. the normal comes from a fixed set of orientations). The intersection tests in the hierarchy can be reduced to interval checks by realigning one of the two collision candidates. The method is very fast (real time even for thousands of polygons) but only suitable for rigid objects.
- The Oriented Bounding Box (OBB) Trees have been presented by [Gottschalk et al. 1996]. Again, the hierarchical structure has been optimized for rigid objects with hundreds of thousands of polygons. A precomputation of the bounding boxes and efficient traversal algorithm allows real time processing of collision detection.

2.2 Distance Fields

A distance field approach is suitable where rigid objects collide with deformable objects for instance. We will discuss the approach described in [Fuhrmann et al. 2003]. The method proposes to represent rigid objects that are likely to interact (i.e. collide) with deformable meshes like fabric or cloth as distance fields. The deformable objects themselves ought to be modeled as particle systems to make the method useful.

The algorithm features two main advantages:

1. Even complex geometry can be represented easily. There is no need for dealing with complex polynoms and approximation

algorithms.

2. The evaluation of normals (and consecutive distances) that are needed for collision detection can be done very fast making the algorithm suitable for real time applications even on standard PCs.

The most important parts of the algorithm are first to build up the distance field and second the actual collision detection process (again, collision response is required but this will not be covered here in detail).

The distance field basically assigns a distance $D(p)$ to every point $p \in \mathbb{R}^3$ which is defined as the minimum distance of p to a point q on the surface of the deformable object. Namely, it is the normal distance to the object. Also, a signum function is applied. Therefore, the mesh needs to be oriented and $sgn(p)$ assigns -1 and 1 to points p laying in front or behind the mesh respectively. For performance reasons, only face normals are used for distance calculations, edge and vertex normals are not used.

The algorithm works as follows: first, prisms are constructed above and below each triangle of the mesh by moving its endpoints in the direction of the face normal and the opposite direction respectively. The height of the prisms is evaluated as delta times the cell diagonal, thus the distance covered can be varied easily. Now axis aligned bounding boxes are constructed around the prisms and subsequently distances and the sign are calculated for every point on the grid inside the bounding boxes using *Voronoi regions* of the features of the triangles. Throughout the successive process of construction it is possible for grid points to change sign. This is not valid in all cases (namely, concave areas are problematic), so these need to be addressed in a special way (see [Fuhrmann et al. 2003] for details).

The distance field is used for the collision detection. In order to demonstrate the accuracy, the original shape of the object can be reconstructed using the *marching cubes* method. Figure 3 illustrates a performed reconstruction of a model of a human's head. The distance between reconstructed and original object has been measured by *Haussdorff distance* and dark blue regions indicate no difference. Since the ears consist of small and concave mesh structures, the distance is greater here. It is sufficient for collision detection, though.

So far, the construction of the distance field has been described. Namely, it is a tight but not minimal envelope around the object that is sufficient for collision detection. Consequently, now the very simple actual collision detection algorithm needs to be applied. Fuhrman et al. decided not to perform a complete collision detection but to test only the vertices for performance reasons (the goal was to create a real-time application). In order to not obtain unpleasing artifacts (interpenetrations) at sharp corners, the objects need to be sepparated at a distance ε.

Based on the assumption that a particle system models the deformable object (namely a set of particles p_i with associated masses m_i) the algorithm works as follows. First, the new position of the particles are computed with *Newton's equation of motion*. External and internal forces are taken into account as well as the initial (i.e. old) velocities of the particles.

The computated new positions are arbitrary, i.e. are generally not lying on the grid at certain grid positions. Therefore, to obtain the approximated distance of the arbitrary position P to the object, trilinear interpolation is performed where the interpolation points are the 8 grid positions surrounding P. The technique has been illustrated in Figure 4. In the figure, the point has been moved to its new position using Newton's equation of motion. Now trilinear interpolation of known grid-point distances is performed to obtain the approximated distance to the grid.

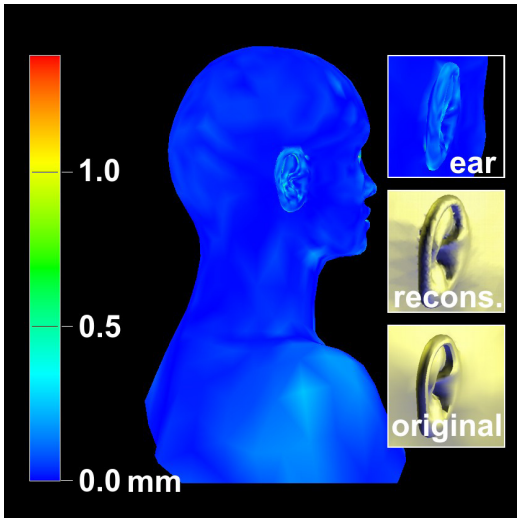

Figure 3: Distance of the reconstruction (performed with marching cubes) of a human's head from the calculated distance field to the original object. The distance is near zero except for the ears where small concave mesh structures lead to greater distances; taken from [Fuhrmann et al. 2003]

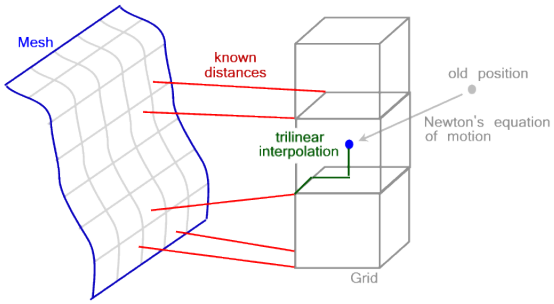

Figure 4: Trilinear interpolation is used to obtain an approximated distance of a point's new location to the mesh.

The collision detection itself is simple: if the distance is below ε, the deformable object is pushed back in its opposite normal direction (the normal is calculated by trilinear interpolation, too). The position change has impact on the velocity, so this needs to be taken into account for further calculations (the collision response is described in more detail in [Fuhrmann et al. 2003]).

A major drawback of the approach is that it does not incorporate self collision detection. Also, all collisions are inelastic, i.e. the energy of the collision is lost. Elastic collisions could be modeled by changing the calculation of the amount r_n what is the scaling factor for the vector in whose direction the deformable object is pushed back. However, the method produces fairly good results at low computational costs for rigid body collisions with deformable objects.

The distance field method could be further enhanced, for instance by using *Adaptive Distance Fields (ADFs)* as proposed in [Frisken et al. 2000]. The method for distance field creation described above might require a lot of memory and unnecessary computation time. Consider an object that features fine detail but only at some posi-

tions: To accurately represent this object by a distance field, the grid has to be very dense, thus resulting in a great amound of grid cells and causing calculation overhead.

[Frisken et al. 2000] proposes the usage of adaptively sampled distance fields by using octrees. This way, the grid samples can be dense where the object shape features details while the sampling rate is reduced where this is not the case. The representation of a human's head (see Figure 3 above) for instance might have an effective representation as adaptively sampled distance field as the ears feature high details while the rest does not.

Figure 5 emphasizes well the advantage of using octree representations for the grid that is used as distance field. Areas where the object does not feature details can have a very low sampling rate while detailed regions are covered to an arbitrary level of detail.

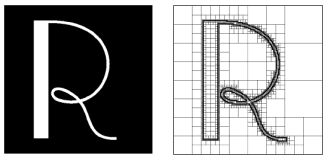

Figure 5: Octree-like subdivision (right) of an object (left) that features detail at only few regions. Memory and computation time can be saved while details are represented to desired level of detail; taken from [Frisken et al. 2000]

The octree however does not only allow adaptive sampling but stores the field in a hierarchical way, too, so besides memory storage savings also computation time can be saved. The distance function $h(x)$ (or $D(p)$ as labeled in the method described above) itself can still be calculated in an arbitrary way; the presented approach only describes the grid creation. To obtain the ADF, a bottom-up or a top-down approach might be used where the latter is recommended as it features a subdivision algorithm which can be controlled by arbitrary predicates that reflect the primary interest of the subdivision.

The adaptively sampled distance field method might work well with the approach of Fuhrman et al. since there are no restrictions for distance function assignments. However, the method has not been tested by either of the two for the domain of deformable objects colliding with rigids.

2.3 Voxel based techniques

A voxel-based method has been introduced by [Chen et al. 2004]. In this approach, collision detection is performed by testing if the voxel-based representation of two objects share same voxels. This is done due to the observations that:

1. objects interfere if and only if their surfaces intersect and
2. this is the case if two objects share at least one common voxel.

To make this work properly, however, the voxelised representations must be accurate enough.

The approach is claimed to work for arbitrary objects, closed or open, rigid or deformable, thus being suitable also for cloth animation. The authors emphasize that the voxelisation process itself is the most demanding task in terms of performance. Therefore, they exploit programmable graphics hardware to achieve interactive frame rates. The authors consider the voxelisation process as their main contribution to the issue.

The basic setup of the method is as follows. Input to the algorithm are two arbitrary objects that ought to be checked for collisions. First, a region of interest (ROI) is calculated using *Axis Aligned Bounding Boxes (AABBs)* that cover the complete objects. By performing simple (and computationally cheap) interval tests, the ROI is determined as the overlapping region of the AABBs (thus, the ROI is an axis aligned cuboid itself). Consequently, voxelisation is performed for both input objects for the parts that lie inside the ROI. Again, those parts can be determined fast using interval tests. The collision detection itself for the two calculated volumes is simple: if the volumes share at least one voxel, collision has occured and needs to be handled. The whole process is embedded into a discrete time step framework as in the other approaches presented in this paper.

The real-time voxelisation process is the bottleneck of this method in terms of computational cost. To handle the process in real time the authors chose to implement the voxelisation on graphics hardware. Intuitively, one would rasterize the mesh into a 3D texture to represent it in a discrete voxel space. At the time (2004), graphics hardware however generally did not support three dimensional textures and in order to make the method available to everyone, Chen et al. decided to use 2D textures instead. These textures generally have four channels per texel (red, green, blue and alpha), each containing 8 bits of information. The authors emphasize that each texel can be used for multiple voxel representations (i.e. for binary voxelisation, the red channel of one texel alone could handle 8 voxels). Thus, a single 4 channel 2048×2048 texture can represent a binary voxelisation of a volume with size 512^3.

The needed texture size, however, could be larger than the graphics card supports. In this case a division into multiple patches is recommended. Namely, a voxelisation that is performed slab by slab along one specified axis direction is to be used (remember, the slabs only need to be of the resolution of the ROI). The slabs might be aligned next to each other in a texture and additional textures ought to be used where necessary. The texture is labeled "worksheet" in the approach and the process is visualized in Figure 6.

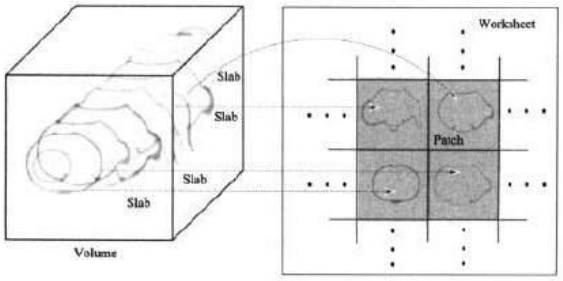

Figure 6: The voxelisation process consists of computing the texel position out of the 3D volume coordinates. The worksheet is organized in patches that represent slices through the volume. Where appropriate (not in the figure), additional textures have to be used (i.e. if the texture size can not cover the whole volume); taken from [Chen et al. 2004]

The actual voxelisation is performed by computing the correct texel position out of the 3D position coordinates for this voxel. To make the calculations more efficient, the texalisation of a triangle is first performed in the direction of the axis that is most parallel to the triangle direction and consequently in the other directions, too. This information is encoded in three directional sheet buffers that are written to the final texture (i.e. the worksheet which represents the volume) once they are filled.

The collision detection is performed by rendering rectangles that are textured with the two textures that represent the voxelisation of the objects. The pixels of these rectangles are compared pairwise by an occlusion-query fragment shader. This way, the total number of colliding voxels as well as the voxels themselves are identified quickly. However, to obtain the affected tringles of the original objects efficiently, origin information should be encoded in the textures. This is no structural problem but not performed by Chen et al. due to performance reasons. The video memory bandwidth did not suffice to reach interactive frame rates using this extension. However, graphics cards performance (and their bandwidth) has been improved significantly since 2004 so today the extension might work in real time.

This assumption has been encouraged by the fact that in their approach, Chen et al. used an ATI 9800Pro 265MB graphics card for experimental calculations of the described voxelisation. Todays high-end graphics cards like the ATI Radeon HD 2900 XT which has been released in May 2007 perform significantly better: Up to three times higher fillrates and bandwidth are reached[1]

In [Chen et al. 2004] no collision response is described so this issue is left to other methods. The presented algorithm however yields colliding voxels effectively in real time and can therefore be expanded well to an interactively working collision prevention framework.

Voxelisation has been used for collision detection by other authors, too. Another approach can be found in [Zhang and Yuen 2000]. In contrast to the method described above, no graphics cards features like buffers or textures are used. Instead, computations are performed in the CPU. The authors, however, describe the voxelisation process in more detail. For example, the optimal voxel size is determined as the size of the longest edge in the mesh (this leads to a voxelisation where each edge can be represented by a maximum of 4 voxels). Also, in this approach the voxels' triangle origin information is stored along with the other voxel data.

Having reduced the number of potentially colliding regions with the voxelisation, the authors also describe a self-collision detection algorithm. A comparable method is not presented in the previously explained approach. Zhang and Yuen however introduce a curvature based approach that is based on the fact that regions of sufficiently low curvature can not intersect themselves (this property can be easily checked by calculating the dot product of the affected triangles' normal vectors).

Basically, the voxelisation method does the following: First, it subdivides the surface (cloth, fabric) into subsurfaces. Those are believed to being able to intersect one another while self-collision of subsurfaces is not checked by the subdivision algorithm. Then, voxelisation is performed and the result used as decision criterion: If two subsurfaces are present in one voxel, collision might occur. This case has to be handled with traditional collision detection afterwards.

Since the method of curvature based collision detection (dot product method) can be used to exclude regions that can not intersect but fails to predict if and where collisions occur in regions of high curvature, it is extended by the *multi-layer concept*. A layer is understood as the small part of a cloth that is going through a particular voxel in this context. Therefore, two layer types exist: single layers (i.e. voxels containing only one layer) and multiple layers (voxels containing more than one cloth layer).

[1] for a comprehensive list of graphics cards and their features see http://www.techarp.com/showarticle.aspx?artno=88 or vendor's product catalogues.

As demonstrated in Figure 7, the surface of the cloth is divided into subsurfaces that are distributed in the voxels. Now the concept of layers becomes important: If all triangles touching a voxel belong to the same subsurface, a single layer case is present, as in Voxel A in Figure 7. In this case, since the subsurfaces are small enough, the curvature method can be used to check whether there are self-collisions.

The other case (presented in voxel B in Figure 7) would be a voxel that contains triangles from different subsurfaces (i.e. multiple layers). This situation might well result in a collision between the present subsurfaces what actually is a self-collision when the fabric is regarded as a whole. Thus, this case makes triangle-wise self-collision tests necessary.

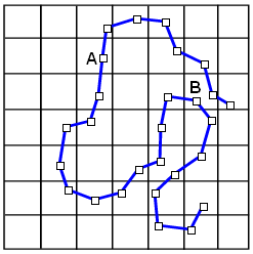

Figure 7: The concept of layers: "A" represents a voxel that contains one single layer so the curvature method can be applied. Voxel "B" contains multiple layers so local collision tests between the subsurfaces need to be performed; drawn after an idea of [Zhang and Yuen 2000]

Note, however, that the necessary tests are reduced to a minimum. With the presented method the voxels are identified that contain potential collisions and these collisions can be handeled locally. [Zhang and Yuen 2000] further introduce an efficient way of identifying the locally small regions of triangles that are used for subsurface generation. The adjacency information contained in the mesh alone would not suffice to make this method work efficiently so a triangle mapping method is introduced.

Besides the self-collision detection, the method proposes a simple algorithm for human body - cloth collision detection. Although this method is not 100 per cent accurate it suffices for realistic rendering and is computationally inexpensive. The authors propose "auxiliary line segments" around the model of the human body. Namely, these line segments are short lines in the direction of the model's normal vectors at its vertices. The length of these segments is one third of the largest cloth edge. Collisions are then assumed to occur if an auxiliary line segment intersects a cloth triangle. This is computational effective since the voxelisation of the cloth, calculated for self-collision, can be used and only the line segments need to be voxelized additionally. Also, no expensive edge-to-edge or vertex-to-triangle checks are necessary. These are all replaced by line-to tringle intersection tests.

2.4 Image based collision detection

Until now a number of object-space based approaches to collision detection in the context of cloth simulation have been presented. The methods aimed for detecting either self-collision or collision with rigid objects or both. A completely different method has been proposed by [Baciu and Wong 2003]. Here, the collision detection is transfered to the image space to bypass the bottleneck of CPU collision detection calculations. I have already presented the approach of [Chen et al. 2004] that makes use of graphics hardware

but this is completely different: while the method decribed earlier only uses textures to represent a voxelisation, Baciu and Wong transfer the whole detection method to the graphics pipeline utilizing ray casting methods. They do, however, use a hybrid approach to identify the region of interest by bounding volumes to make calculations even more efficient.

The method is inspired by the observation that traditional object-space approaches share common weaknesses. Among them are:

- When the geometry in the scene becomes more complex (i.e. more or more detailed objects), this has great impact on the performance of the methods, regardless of their efficiency.
- All methods need complex data structures (i.e. hierarchies like octrees, bounding volumes, voxelisation representation) that have to be precomputed. Most structures need to be updated regularly.
- Special cases are present in most scenes that need to be taken into account, slowing the whole process down.

The proposed method basically makes use of *ray casting* and tries to identify object interference by depth values. Ray casting itself can be seen as the projection of the three dimensional space onto the viewing plane, thus resulting in a sequence of images when depth information is taken into account. It is intuitively clear, that interference can occur if and only if (a) the projected regions of two objects A and B overlap (i.e. their extension in the direction of the viewing plane) and (b) their depth intervals $I(A)$ and $I(B)$ overlap (i.e. the extension in z direction). Conrete cases are discussed later, first the setup of the region that has to be tested is discussed.

The authors introduce the term *Minimum Overlapping Region (MOR)* as the region of interest. It is calculated as the rectangular region that overlaps both projections of the objects' A and B axis aligned bounding boxes (AABBs). Consequently, the MOR is used as the source plane from where rays are cast orthogonaly into the scene. Therefore, it is subdivided into grid cells.

The scene setup is presented in Figure 8. The figure shows (on the right side) two objects and their assigned bounding boxes. The viewing plane shows the projections of the AABBs and the resulting MOR.

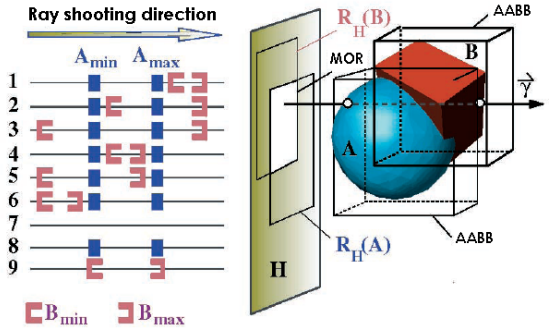

Figure 8: (left) possible cases for depth ordering of two objects for a ray cast into the scene; (right) the setup of the scene, including the objects, the AABBs and the viewing plane with the MOR; taken from [Baciu and Wong 2003]

The setup of the MOR and the ray casting can be seen as the process of defining a viewport and performing orthogonal projection in classic computer graphics terminology. The correct viewport (i.e. MOR) setup is crucial for performance since additional collision checks (necessary for each pixel of the viewport) are computationally expensive.

Once the setup is complete and depth tests can be performed, collision can be detected with respect to a set of rules that is presented graphically on the left side of Figure 8. The figure shows the maximum and minimum depth values of the objects A and B, being Z^A_{min}, Z^A_{max} and Z^B_{min}, Z^B_{max} respectively. The four values can have a total of $\frac{4!}{2!*2!}+3=9$ permutations (the $+3$ originate from the trivial cases (7) no object (8) only object A and (9) only object B) and are illustrated in Figure 8. The figure emphasizes that no collision occurs only in the cases 1, 6, 7, 8 and 9 and that collision detection can effectively be performed by calculating the minimum and maximum depth values of the objects and comparing them to one another.

The computations are performed on graphics cards that feature a depth and stencil buffer. However, since there is only one depth buffer, interval tests can not be performed straight forward. At any given time, only one depth value per pixel can be accessed (namely, the Z_{max} values of the just rendered object). To deal with this inconvenience, the authors chose the following procedure: Being not able to gather all four necessary values at the same time, they perform the test

$$Z^A_{max} > Z^B$$

where Z^B is the depth value of object B and count the number of passes for which it holds pixel-wise. The result (per pixel) is stored in the stencil buffer, that is, a two dimensional array $SC(p)$ of exactly the same dimension as the viewport.

There are three possible counts that are written into $SC(p)$: 0 indicates that at pixel p, object B is completely behind object A or that B does not overlap this pixel. The value $SC(p)=1$ is obtained when the objects A and B collide with each other (cases 2 or 3 in Figure 8). However, the count $SC(p)=2$ is ambiguous. It indicates that both Z^B values are smaller than Z^A_{max} but this could mean (1) that the objects collide or (2) that B is located completely in front of A and the two do not collide. The situation is illustrated as case 6, 7 and 8 in Figure 8. To overcome this difficulty, the test is expanded in the case of a stencil buffer count of 2. Therefore, the objects are rendered in reversed order (i.e. the symbols A and B are switched). Now case 4 becomes case 3, case 5 becomes case 2, and case 6 becomes case 1. These cases (1, 2, 3) can be solved without any further processing so altogether a maximum of two rendering passes is necessary to perform the complete collision detection.

This, however, only holds for convex objects. The authors emphasize that the processing of deformable objects needs no further adjustments while in the case of concave objects (this is, as the deformation issue, likely in cloth simulation), generally an unbounded number of passes is necessary (for an unbounded number of folds in one direction). Even cloth, though, does not have an unbounded number of folds along one viewing ray so the number of necessary paths is reduced.

The technique is further refined by calculating optimal MORs with tight AABBs and by using a *separating vector algorithm* for initial decisions about potentially colliding objects. Details about this processes can be found in [Baciu and Wong 2003]. Also, detailed results and comparison with other collision detection algorithms are covered within the paper.

2.5 Other methods

There exist other methods that are not decribed here due to space and time limitations. Among them we find:

- Stochastic methods that offer tradeoffs between accuracy and speed.
- Spatial partitioning in general (distance fields or AABBs can be seen as special cases of spatial partitioning)
- Other hardware based methods with the main disadvantage that since calculations are performed in image space, the resolution is limited to this space. This can result in missed collisions between small triangles due to sampling errors.
- Other methods like the chormatic decomposition of cloth meshes using graph coloring as presented by [Govindaraju et al. 2005].

3 Summary and Conclusion

Collision detection has become a demanding task as models get more detailed and complex. By aiming for steadily improved realism, the computer graphics society models not only rigid objects but also fabrics for their scenes in steadily better resolution, thus resulting in more triangles or other primitives for the mesh.

The modeling of cloth induces several specialties. In section 1 I introduced some of them in more detail:

- The decision about the mesh structure. Common are triangle meshes, but collision detection has been performed on time-dependent parametric surfaces, on subdivision surfaces and even on meshes represented by particle systems.
- The modeling of internal cloth dynamics. I introduced the popular mass-spring model that is well described in [Louchet et al. 1995] and has been used by the presented method of [Bridson et al. 2002].
- The rendering issue that is crucial for realistic, "look and feel" preserving cloth simulation

In section 2 I presented a number of classical approaches to collision detection. Some of them have been designed in particular for deformable surfaces or cloth, while others have been found to work for these domains, too (possibly with some changes or extensions) although they have been developed for rigid object interference detection.

As the ongoing prosperity of academic papers in the field of collision detection for deformable objects lets presume, the work is a long way off of finding ideal solutions to the problem. Solutions exist for special cases, though.

Some tasks might not require real-time functionality. This is present mainly in the film and animation industry. Here, accurate collision detection of high quality needs to be performed while rendering time is not crucial (in fact, for modern animation films, single frames are rendered for several hours). For this domain, the presented approach of [Bridson et al. 2002] might suit as it claims to have guaranteed no self-interference of cloth.

In other areas real-time is crucial, like in the games industry. The developers need to find tradeoffs between realism and performance and until now hierarchical bounding object approaches seem to work well. While the precision is not as high, performance can be controlled well by setting the maximum number of layers in the hierarchy.

Yet another domain is the area of interactive product catalogues for cloth where realism and real-time performance are both important. However, in these settings there is normally not much movement or object interference involved so collision detection can be reduced to a minimum.

Said that, the community is still aiming for more efficient and more accurate algorithms. The image-space approaches have brought the focus on graphics hardware. Being not able to handle complex concave objects in reasonable time at the moment, this might change as the hardware features enhance. Multiple depth buffers and easier storage of values would certainly improve the cloth rendering ability, also in terms of collision detection.

References

BACIU, G., AND WONG, W. 2003. Image-based techniques in a hybrid collision detector. *Visualization and Computer Graphics, IEEE Transactions on 9*, 2, 254–271.

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

CHEN, W., WAN, H., ZHANG, H., BAO, H., AND PENG, Q. 2004. Interactive collision detection for complex and deformable models using programmable graphics hardware. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, New York, NY, USA, 10–15.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 249–254.

FUHRMANN, A., SOBOTTKA, G., AND GROSS, C. 2003. Distance fields for rapid collision detection in physically based modelling. In *Proceedings of GraphiCon 2003*.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. Obbtree: a hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 171–180.

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. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 991–999.

GRINSPUN, E., CIRAK, F., SCHROEDER, P., AND ORTIZ, M., 2001. Non-linear mechanics and collisions for subdivision surfaces.

HERZEN, B. V., BARR, A. H., AND ZATZ, H. R. 1990. Geometric collisions for time-dependent parametric surfaces. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 39–48.

JAMES, D. L., AND PAI, D. K. 2004. Bd-tree: output-sensitive collision detection for reduced deformable models. In *SIGGRAPH*

'04: ACM SIGGRAPH 2004 Papers, ACM Press, New York, NY, USA, 393–398.

LOUCHET, J., PROVOT, X., AND CROCHEMORE, D. 1995. Evolutionary identification of cloth animation models. In *Computer Animation and Simulation '95*, Springer-Verlag, D. Terzopoulos and D. Thalmann, Eds., 44–54.

MOORE, M., AND WILHELMS, J. 1988. Collision detection and response for computer animation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 289–298.

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2003. Efficient and realistic visualization of cloth. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 167–177.

VOLINO, P., COURSHESNES, M., AND MAGNENAT THALMANN, N. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH 95 Conference Proceedings*, Addison Wesley, R. Cook, Ed., 137–144.

ZACHMANN, G. 1998. Rapid collision detection by dynamically aligned dop-trees. In *Virtual Reality Annual International Symposium, 1998. Proceedings IEEE 1998, Vol., Iss., 14-18 Mar 1998*, IEEE press, 90–97.

ZHANG, D., AND YUEN, M. M. F. 2000. Collision detection for clothed human animation. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 328.