# A Multi-Scale Singularity Bounding Volume Hierarchy

Somchaipeng, Kerawit; Erleben, Kenny; Sporring, Jon

*Publication date:*
2004

*Document version*
Early version, also known as pre-print

Datalogisk Institut
KØBENHAVNS UNIVERSITET


# A Multi-Scale Singularity Bounding Volume Hierarchy

**Kerawit Somchaipeng, Kenny Erleben and Jon Sporring**

**Technical Report no. 2004/08**
**ISSN: 0107-8283**
**CR Subject Classification: I.3, I.6, I.4**

**DIKU**
University of Copenhagen • Universitetsparken 1
DK-2100 Copenhagen • Denmark

# A Multi-Scale Singularity Bounding Volume Hierarchy

K Somchaipeng [1,2], K. Erleben [2], and J. Sporring [2]

[1]3D-Lab, School of Dentistry, Dept. of Pediatric Dentistry, University of Copenhagen Noerre Alle 20 DK-2200 Copenhagen N. Denmark

[2]Dept. of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100, Copenhagen N, Denmark

**Abstract**

A scale space approach is taken for building bounding volume hierarchies for collision detection. A spherical bounding volume is generated at each node of the bounding volume hierarchy using estimates of the mass distribution.

Traditional top-down methods approximates the surface of an object in a coarse to fine manner, by recursively increasing resolution by some factor, e.g. 2. The method presented in this article analyzes the mass distribution of a solid object using a well founded scale-space based on the Diffusion Equation: the Gaussian Scale-Space. In the Gaussian Scale-Space, the deep structure of extremal mass points is naturally binary, and the linking process is therefore simple.

The main contribution of this article is a novel approach for constructing bounding volume hierarchies using multi-scale singularity-trees for collision detection. The bounding volume hierarchy building algorithm extends the field with a new method based on volumetric shape rather than statistics of the surface geometry or geometrical constructs such as medial surfaces.

## 1 Introduction

In physics-based animation, collision detection often becomes the bottleneck, since a collision query needs to be performed in every simulation step in order to determine contacting and colliding objects. Animations can have many objects, all of which may have a complex geometry, such as polygonal soups of several thousands facets, and it is therefore a computationally heavy burden to perform collision detection especially for real-time interaction.

Bounding volume hierarchies are widely used in computer graphics, e.g. for ray tracing [14], and they are quite popular in animation (e.g. [5] uses them for cloth animation), since they are applicable of handling more general shapes

than most feature-based and simplex-based algorithms, they tend to generate smaller hierarchies than spatial subdivision algorithms, and they offer a graceful degradation of objects, which is highly useful when accuracy is to be traded for performance. New performance improvements of bounding volume hierarchies is therefore of great practical and theoretical interest to the computer graphics and animation community.

The main contribution of this paper is a novel algorithm for bottom-up construction of spherical approximating bounding volume hierarchies. We prefer our hierarchies, firstly because they save memory, and therefore increases simulation performance, when compared to traditional bounding volume hierarchy, and secondly because they are a direct implementation of the mass of objects rather than their boundary representation.

In this article we will restrain ourselves from the n-body problem and only consider narrow phase [21] collision detection of solid non-deformable objects.

## 1.1 Previous Work

There is a wealth of literature on collision detection, and many different approaches have been investigated. Spatial subdivision algorithms like Binary Space-Partitioning (BSP) tree [32], octree [44, 12, 11], k-d trees and grids [12, 11], feature-based algorithms like polygonal intersection [34], Lin-Can [40], VClip [33], SWIFT [10], recursive search methods [43], simplex-based such as GJK [13, 45], generalized Voronoi diagrams [20], and signed distance maps [17, 5, 19]. Finally there are algorithms based on bounding volume hierarchies such as ours.

Bounding volume hierarchies have been around for a long time. Consequently there is a huge wealth of literature about bounding volume hierarchies. Most of the literature addresses homogeneous bounding volume hierarchies and top-down construction methods. A great variety of different types of bounding volumes have been reported: Spheres [22, 37, 9], axed aligned bounding boxes (AABBs) [1, 30], oriented bounding boxes (OBBs) [15, 16], discrete orientation polytypes (k-DOPs) [25, 52], Quantized Orientation Slabs with Primary Orientations (QuOSPOs) [18], Spherical shell [27], and swept sphere volumes (SSVs) [29]. In general, it has been discovered that there is a trade-off between the complexity of the geometry of a bounding volume and the speed of its overlap test and the number of overlap tests in a query.

In contrast to bounding volumes types, there has only been written little on approximating bounding volume hierarchies. To our knowledge [21] pioneered the field, where octrees combined with simulated annealing were used to construct a sphere tree, followed by [38, 37], cumulating with a superior bottom-up construction method based on medial surface (M-reps) [22]. More recently [36, 9] used approximating sphere-trees built in a top down fashion based on an octree for time critical collision detection, and [3] used an adaptive m-rep approximation-based top-down construction algorithm.

There have been written even less about heterogeneous bounding volume hierarchies, although object hierarchies of different primitive volume types are

a widely used concept in most of todays simulators [35, 49, 24]. The SSVs [29] are one of the most recent publications. The general belief is, however, that heterogeneous bounding volumes does not change the fundamental algorithms, but merely introduces a raft of other problems. It is also believed that heterogeneous bounding volumes could provide better and more tightly fitting bounding volumes resulting in higher convergence towards the true shape volume of the objects. This could mean an increase in the pruning capabilities and a corresponding increase in performance.

Most of the work with bounding volume hierarchies has addressed objects that are represented by polygonal models. Many experiments also indicate that OBBs (and other rectangular volumes) provide the best convergence for polygonal models [15, 16, 52, 29], while spherical volumes are believed to converge best towards the volume. The underlying query algorithms for penetration detection, separation distance and contact determination of bounding volume hierarchies have not changed much. In its basic form, these algorithms are nothing more than simple traversals.

To our knowledge, the trees based on the deep structure of Gaussian Scale-Space has not been used previously for generating bounding volume hierarchies or collision detection. An alternative to Gaussian scale-space is curvature scale-spaces, from which M-reps are derived. M-reps based methods are state of the art for bottom-up construction method [22] and top-down construction [3]. For deformable objects such as cloth, bottom-up construction based on mesh topology [47, 48, 4] are the preferred choice. In [1] a median based top-down method was proposed for building an AABB tree. [30] suggested using a mesh connectivity-tree in a top-down construction method.

## 2 Creating Hierarchies from Gaussian Scale-Space

The $N+1$ dimensional Gaussian scale-space, $L : \mathbb{R}^{N+1} \to \mathbb{R}$, of an $N$ dimensional image, $I : \mathbb{R}^N \to \mathbb{R}$, is an ordered stack of images, where each image is a blurred version of the former [23, 51, 26]. The blurring is performed according to the diffusion equation,

$$\partial_t L = \nabla^2 L, \tag{1}$$

where $\partial_t L$ is the first partial-derivative of the image in the scale direction $t$, and $\nabla^2$ is the Laplacian operator, which in 3 dimensions reads $\partial_x^2 + \partial_y^2 + \partial_z^2$. An example of the scale-space of a solid cow is shown in Figure 1.

The Gaussian kernel is the Green's function of the heat diffusion equation, i.e.

$$L(\cdot; t) = I(\cdot) \otimes g(\cdot; t), \tag{2}$$

$$g(x; t) = \frac{1}{(4\pi t)^{N/2}} e^{-x^T x/(4t)}, \tag{3}$$

where $L(\cdot, t)$ is the image at scale $t$, $I(\cdot)$ is the original image, $\otimes$ is the convolution operator, $g(\cdot; t)$ is the Gaussian kernel at scale $t$, $N$ is the dimensionality

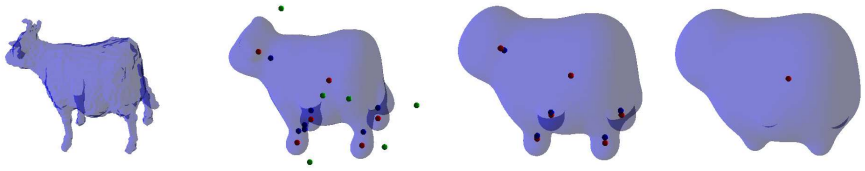


Figure 1: An example of the scale-space of a solid cow [2]. The images show the zero iso-surfaces of the cow at scales $\sigma = 0, 2, 3.75, 5$.

of the problem, and $t = \sigma^2/2$, using $\sigma$ as the standard deviation of the Gaussian kernel. The *Gaussian scale-space* is henceforth called the scale-space in this article. The information in scale-space is logarithmically degraded, the scale-parameter is therefore often sampled exponentially using $\sigma = \sigma_0 e^T$. Since differentiation commutes with convolution and the Gaussian kernel is infinitely differentiable, differentiation of images in scale-spaces is conveniently computed,

$$\partial_{x^n} L(\cdot; t) = \partial_{x^n} \left( I(\cdot) \otimes g(\cdot; t) \right) = I(\cdot) \otimes \partial_{x^n} g(\cdot; t). \tag{4}$$

Alternative implementations of the scale-space are multiplication in the Fourier Domain, finite differencing schemes for solving the heat diffusion equation, additive operator splitting [50], and recursive implementation [8, 46]. Each method has its advantages and disadvantages, we prefer the spatial convolution, since it is guaranteed not to introduce new extrema in homogeneous regions. Typical border conditions are Dirichlet, Cyclic repetition, and Neumann boundaries. We use Dirichlet boundaries, where the image is extended with zero values in all directions.

Although the dimensionality of the constructed scale-space is one higher than the dimensionality of the original image, *critical points*, in the image at each scale are always points. A critical point is e.g. an extremum, $\partial_x L = \partial_y L = \partial_z L = 0$. Critical points are classified by the eigenvalues of the Hessian matrix, the matrix of all second derivatives, computed at the point. Critical points with all positive eigenvalues are minima, critical points with all negative eigenvalues are maxima, and critical points with a mixture of both negative and positive eigenvalues are saddles. In three dimensional space, there are two groups of saddles: one has two positive and one negative eigenvalues and the other has two negative and one positive eigenvalues.

As we increase the scale parameter, the critical points move smoothly forming *critical paths*. Along scale, critical points meet and annihilate or are created. Such events are called catastrophe events, and the points where they occur are called catastrophe points and the collection of events is called the *deep structure* of the image. The notion of genericity is used to disregard events that are not likely to occur for typical images, i.e. generic events are stable under slight perturbation of the image. There are only two types of generic catastrophe

events in scale-space namely pairwise *creation events* and *annihilation events* [7], and it has further been shown that generic catastrophe events only involves pairs of critical points where one and only one eigenvalue of the Hessian matrix changes its sign, e.g. the annihilation of a minimum (+, +, +) and a saddle (+, +, -). A detailed discussion of a robust method for extracting critical paths and catastrophe points from 2+1D and 3+1D scale-spaces can be found in [42].

## 2.1 Extrema-Based Multi-Scale Singularity Trees

There are a few methods for constructing tree structures from the deep structure of two-dimensional images proposed in the literatures so far [31, 28, 39].

To our knowledge, an attempt to construct tree structures from the deep structure of three-dimensional images is first proposed in [42]. The scheme produces rooted ordered binary trees called extrema-based multi-scale singularity trees with catastrophe points as nodes. In three-dimensional space, catastrophes are also possibly caused by creations or annihilation of saddle-saddle points, e.g. between (+, +, -) and (+, -, -) saddles, corresponding to the catastrophic interaction between a thinning and a thickening of a tube. These catastrophes together with all creation catastrophes are ignored, since we are only interested in the linking of the extrema which present at the lowest scale.

The binary tree structure is obtained by linking catastrophe points, and the algorithm is as follows: Let $\Omega \subset \mathbb{R}^3$ be a compact connected domain and define $I : \Omega \to \mathbb{R}$ to be an image, $\vec{e} \in \Omega$ as an extremum, and $\vec{x} \in \Omega$ as a point. Consider the set of continuous functions $\gamma : [0, P] \to \Omega$ for which $\gamma(0) = \vec{e}$, $\gamma(P) = \vec{x}$, and $\gamma \in \Gamma_{\vec{e}\vec{x}}$, where $\Gamma_{\vec{e}\vec{x}}$ is the set of all possible paths from an extremum $\vec{e}$ to a point $\vec{x}$, and $\gamma$ is parameterized using Euclidean arc-length. The energy $E_{\vec{e}}(\vec{x})$ with respect to an extremum $\vec{e}$ calculated at $\vec{x}$ is defined as

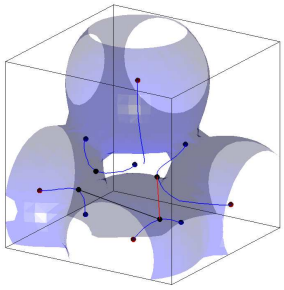
$$E_{\vec{e}}(x) = \inf_{\gamma \in \Gamma_{\vec{e}\vec{x}}} \int_0^P \sqrt{\left| x(\vec{\gamma(p)}) \right|^2 + \nu \left( \frac{dI(\gamma(p))}{dp} \right)^2} \, dp. \tag{5}$$

Let $\mathcal{E} \subset \Omega$ be the set of all extrema in the image. The *extrema partition*, $Z_i$, associated with an extrema $\vec{e}_i \in \mathcal{E}$ is defined as the set of all points in the images, where the energy $E_{\vec{e}_i}(\vec{x})$ is minimal,

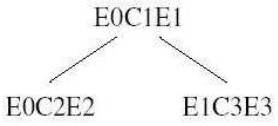
$$Z_i = \left\{ \vec{x} \in \Omega \middle| E_{\vec{e}_i}(\vec{x}) < E_{\vec{e}_i}(\vec{x}), \forall \vec{e}_j \in \mathcal{E}, i \neq j \right\}. \tag{6}$$

An approximation of the energy map $M_i : \Omega \to \mathbb{R}^+$, which defines the energy at every point in the image associated with an extremum $\vec{e}_i$, can be efficiently calculated using the *Fast Marching Methods* [41].

At an catastrophe point, one extrema disappears, implying that also one extrema partition vanishes. The algorithm then constructs the tree structure based on the nesting of extrema partitions across scales. At a catastrophe, a node consisting of a catastrophe and two ports is created. The left port will be a reference to an extremum whose extrema partition is present immediately after the catastrophe in scale, and the right node will be the extremum that is lost



(a)




(b)

Figure 2: The extrema-based multi-scale singularity tree of a three-dimensional image of four blobs. (a) The 2.0 iso-surfaces of the image at scale $\sigma = 3$ is shown in blue, where the scale axis is projected away. The small red and blue spheres are the maxima and saddles points respectively. The blue lines are the critical paths and the small black spheres are the catastrophe points. The black line and the red line denote the left-child linking and the right-child linking in the tree. (b) A schematic drawing of the extracted MSST. Note that there is a saddle-saddle catastrophe point which is ignored.

at the catastrophe. The nodes are linked in a top-down manner, starting from the highest to the lowest catastrophe in scale. There will be as many nodes as there are catastrophes in the constructed tree. A Full discussion of the method for constructing extrema-based MSSTs can be found in [42].

An example of the extrema-based multi-scale singularity tree of a simple 4-blob 3D image is shown in Figure 2.

## 2.2 Bounding Volume Hierarchy From a Multi-Scale Singularity-Tree

Given a multi-scale singularity-tree, we produce a Bounding Volume as follows. The multi-scale singularity-tree is extended with a set of leaves representing each extremum. The newly added leaves represent the finest scale for the Bounding Volume Hierarchy. The nodes of the multi-scale singularity-trees consists of two ports, a left port referring to the extremum surviving the catastrophe, and the right port referring to the extremum that is annihilated. All free ports are extended with a leaf for the corresponding extremum, and the result is that the multi-scale singularity-tree is extended with one leaf for each extremum. All except the last extremum will appear in the multi-scale singularity-tree one and only one time as a right port.

We can denote the catastrophe scale as a size measure of the corresponding extremum. That is, at the catastrophe scale, the corresponding Gaussian will

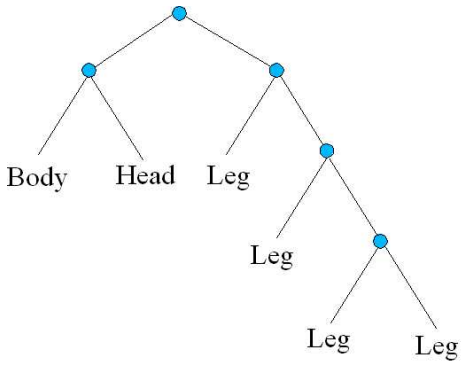



Figure 3: A schematic drawing of the extracted BVH of a solid cow.

have a size that dominates the underlying image structures. We may also give a statistical interpretations using Tchebycheff's inequality [6]. It states that for a random variable $X$ with standard deviation $\sigma$, the probability of finding values outsize a bound proportional to its standard deviation is inversely small:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{7}$$

We take this as a guide to set the size of the leaf bounding volumes, i.e. a leaf will be given a sphere, who's radius is proportional to the catastrophe scale. There will be one extremum, which does not disappear in a catastrophe, and therefore does not appear as a right port in the multi-scale singularity tree. We set the bounding volume of the final extremum to be proportional to the distance to its only sibling in the multi-scale singularity-tree minus the already known sibling's radius in the bounding volume hierarchy.

Since the bounding volume hierarchy is binary, we find bounding volume for the non-leaf nodes in the tree as the smallest sphere that encloses the two child spheres. Although tighter bounds may be found, this is left for further development.

# 3 Results

Currently, our algorithm is capable of producing trees from objects that are sampled on a $256^3$ grid, for a reasonable computation time, we only use $64^3$ grids. We demonstrate our algorithm on the cow polygonal mesh [2]. In Figure 3 is shown a schematic drawing of the extracted BVH of a solid cow and in Figure 4 is shown a solid cow together with the spherical bounding volume at each level.

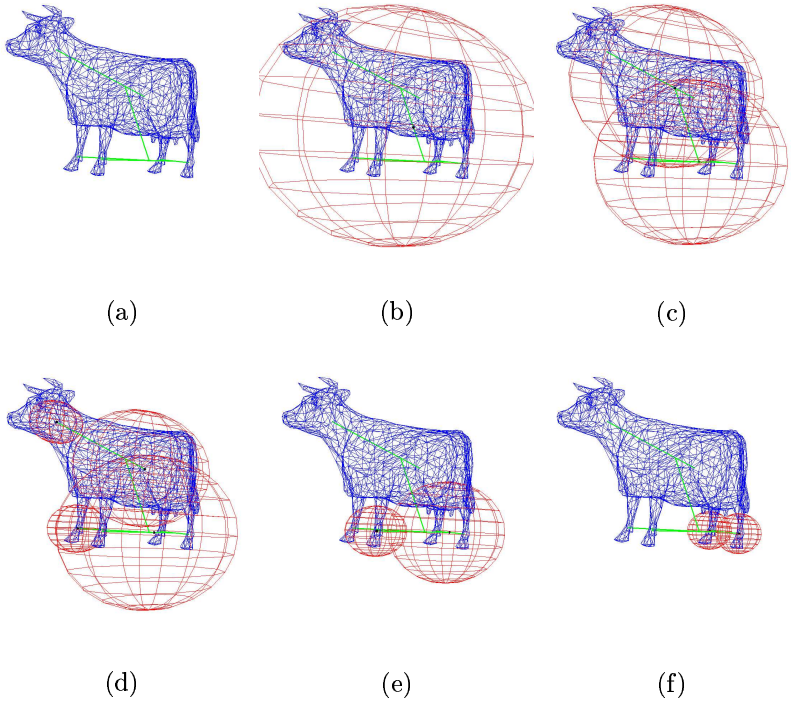



Figure 4: A solid cow and the levels in its bounding volume hierarchy. In (a) is shown the surface of the cow together with the links between catastrophes in the tree. In (b)-(f) are shown nodes at levels 1-5 of the tree.

Due to the nature of catastrophic events in scale-space, the produced multi-scale singularity-tree are always binary trees. In the scale-space of the cow, the legs of the cow appears in a sequential manner from coarse to fine. This makes the tree building process simple, however, in this particular example, it would possibly be more natural to let the leg-nodes appear at the same time in a 4-ary tree node. In our tree, such decisions can be enforced by post-processing, and a useful indication in this case would be that the catastrophes occur within a very narrow scale-band.

There are many properties which are interesting when evaluating the quality of a bounding volume hierarchy. Unfortunately some of them are contradicting each other.

- Smallest possible bounding volumes
- Smallest possible overlap between volumes at same depth in the hierarchy
- Small sized bounding volume hierarchy, i.e. as few nodes as possible
- Complete coverage versus sampling based coverage
- "balanced" trees

The last property is one we challenge, although it has been proved that balanced trees provide best worst case queries, a balanced tree do not represent the scale of the object. Working with time critical or approximating queries this become an important property. We suggest that the tree should be balanced with respect to the density of the object.

# 4 Discussion

Most recent work with bounding volume hierarchies has focused on: Trying out new kinds of bounding volumes, figuring out better methods for fitting a bounding volume to a subset of an object's underlying geometry, finding faster and better overlap test methods, and comparing homogeneous bounding volume hierarchies of different bounding volume types. In order to improve the performance of traversal algorithms, depth control, layered bounding volumes, caching bounding volumes, and shared bounding volumes have been studied. We have chosen to classify our method as being a mixed bottom-up and top-down method, because the scale-space is build bottom-up, and the multi-scale singularity tree are found in a top-down manner. The corresponding bounding volume hierarchy is then built in a straightforward incremental way, by doing an order traversal of the multi-scale singularity tree, and creating bounding volume nodes as catastrophes are encountered.

The computational complexity for our algorithm is currently high. Using $N^3$ as the number of pixels in the image, $S$ as the number of scales to be evaluated, $\sigma$ as the largest scale, $K$ as the number of critical line-pieces found, and $E$ as the number of extrema at the lowest scale, the computational complexity for each part of our algorithm is as follows:

**Computation of the Gaussian Scale-Space:** $\mathcal{O}(S\sigma^3 N^3)$

It may be possible to improve the calculation time for the Gaussian scale-space, since many alternatives to spatial convolution are faster for large scales. However, we have not yet found alternatives that does not introduces spurious extrema in homogeneous regions.

**Storage Requirement for the Scale-Space:** $\mathcal{O}(2N^3)$

The most memory intensive part of our algorithm is the storage of the scale-space. We only require the storage of two neighboring scales in order to find the critical paths in our current implementation.

**Extracting Critical Paths:** $\mathcal{O}(SN^3 + K^2)$

The critical paths can be extracted considerably faster by tracking each extremum from finest scale, however this would require either to store the full Scale-Space or perform local calculations during the tracking process. Since this is by far not the slowest part of our algorithm, we have left this for further research.

**Finding a Euclidean Tree,** $\nu = 0$ **in (5):** $\mathcal{O}(E^2)$

It is fastest to use the Euclidean metric in (5), for $\nu \neq 0$ see below.

**Finding a General Tree:** $\mathcal{O}(EN^3 \log N^3)$

This is the most computationally expensive part of our algorithm, and we have therefore preferred the Euclidean metric. However, we expect that the speed of the Fast Marching Method used for the alternative metrics, $\nu \neq 0$, can be improved by a narrow band implementation.

Gaussian scale space provides us with a continuous degradation of an object, other algorithms fail completely on this point, they typical control their scale by saying that at next level of the bounding volume hierarchy should have 50% less number of volumes, or at the next level the volumes should fit 20% better. A direct study of scales seems to be a more proper representation.

Medial surface (M-reps) based methods for building bounding volume hierarchies have been the approach to use for bottom-up construction. Our method differs from M-reps significantly by being a density based method, whereas M-reps is more a surface-based method. Furthermore our method provides us with a natural scale that is easily used to determine both bounding volumes and the topology of the hierarchy. M-reps do not provide this scale information nor can they tell one about the density of an object.

The well-established foundation on scale-space theory provides us with a well-defined concept of scale, shape, and detail of an object. These concepts are valuable tools as our work hopefully demonstrates.

The main contribution of our work is a new method for building bounding volume hierarchy, however, there is still much need to be done. So far our work has been a feasibility study showing that the construction actually can be done. We have not yet made any attempt towards comparing the quality of the multi-scale singularity bounding volume hierarchy with other algorithms.

Future research will be on the tightening of the bounding volumes utilizing information in scale-space.

# References


[1] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[2] Gareth Bradshaw. Polygonal mesh of a cow. .

[3] Gareth Bradshaw and Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics*, 23(1), January 2004.

[4] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *Proceedings of ACM SIGGRAPH*, 21(3):594–603, 2002.

[5] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36. Eurographics Association, 2003.

[6] L. Brøndum and J. D. Monrad. *Statistik I - Sandsynlighedsregning og statistiske grundbegreber*. Den private ingeniørfond, 1993.

[7] James Damon. Local Morse theory for Gaussian blurred functions. In Jon Sporring, Mads Nielsen, Luc Florack, and Peter Johansen, editors, *Gaussian Scale-Space Theory*, chapter 11, pages 147–163. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[8] R. Deriche. Recursively Implementing the Gaussian and its Derivatives. In V. Srinivasan, Ong Sim Heng, and Ang Yew Hock, editors, *Proceedings of the 2nd Singapore International Conference on Image Processing*, pages 263–267. World Scientific, Singapore, 1992.

[9] John Dingliana and Carol O'Sullivan. Graceful degradation of collision handling in physically based animation. *Computer Graphics Forum*, 19(3), 2000.

[10] Stephan A. Ehmann and Ming C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 500–510. Blackwell Publishing, 2001.

[11] Kenny Erleben and Jon Sporring. Collision detection of deformable volumetric meshes. In Jeff Lander, editor, *Graphics Programming Methods*. Charles River Media, 2003.

[12] Fabio Ganovelli, John Dingliana, and Carol O'Sullivan. Buckettree: Improving collision detection between deformable objects. In *Spring Conference in Computer Graphics (SCCG2000)*, pages pp. 156–163, Bratislava, April 2000.

[13] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4:193–203, 1988.

[14] Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, May 1987. see Scherson & Caspary article for related work.

[15] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical Report TR96-013, Department of Computer Science, University of N. Carolina, Chapel Hill, 8, 1996.

[16] Stefan Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, Department of Computer Science, University of N. Carolina, Chapel Hill, 2000.

[17] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Transaction on Graphics, Proceedings of ACM SIGGRAPH*, 2003.

[18] Taosong He. Fast collision detection using quospo trees. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 55–62. ACM Press, 1999.

[19] Gentaro Hirota. *An Improved Finite Element Contact Model for Anatomical Simulations*. PhD thesis, University of N. Carolina, Chapel Hill, 2002.

[20] Kenneth E. Hoff, III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press/Addison-Wesley Publishing Co., 1999.

[21] P. M. Hubbard. Interactive collision detection. In *Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality*, pages 24–32, 1993.

[22] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[23] T. Iijima. Basic theory on normalization of a pattern (in case of typical one-dimensional pattern). *Bulletin of Electrotechnical Laboratory*, 26:368–388, 1962. (in Japanese).

[24] Karma. MathEngine Karma, http://www.mathengine.com/karma/.

[25] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of $k$-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[26] J. J. Koenderink. The Structure of Images. *Biological Cybernetics*, 50:363–370, 1984.

[27] S. Krishnan, A. Pattekar, M. Lin, and D. Manocha. Spherical shell: A higher order bounding volume for fast proximity queries. In *Proc. of Third International Workshop on Algorithmic Foundations of Robotics*, pages 122–136, 1998.

[28] Arjan Kuijper. *The deep structure of Gaussian scale space images.* PhD thesis, Image Sciences Institute, Institute of Information and Computing Sciences, Faculty of Mathematics and Computer Science, Utrecht University, 2002.

[29] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill, 1999.

[30] Thomas Larsson and Tomas Akenine-Möller. Collision detection for continuously deforming bodies. In *Eurographics*, pages 325–333, 2001.

[31] L. M. Lifshitz and S. M. Pizer. A multiresolution hierarchical approach to image segmentation based on intensity extrema. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(6):529–541, 1990.

[32] Stan Melax. Bsp collision detection as used in mdk2 and neverwinter nights. *Gamasutra*, March 2001. Online article.

[33] Brian Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, July 1998.

[34] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Computer Graphics*, volume 22, pages 289–298, 1988.

[35] Ode. Open Dynamics Engine, http://q12.org/ode/.

[36] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees, 1999.

[37] I.J. Palmer. Collision detection for animation: The use of the sphere-tree data structure. In *The Second Departmental Workshop on Computing Research.* University of Bradford, June 1995.

[38] I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995.

[39] B. Platel. Multiscale Hierarchical Segmentation. Technical Report BMT-Report no. 2002-04, Department of BioMedical Engineering, Technical Universitet of Eindhoven, 2002.

[40] M. K. Ponamgi, D. Manocha, and M. C. Lin. Incremental algorithms for collision detection between polygonal models:. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):51–64, /1997.

[41] J. A. Sethian. Fast Marching Methods. *SIAM Review*, 41(2):199–235, 1999.

[42] K. Somchaipeng, J Sporring, S. Kreiborg, and P. Johansen. Software for Extracting Multi-Scale Singularity Trees. Technical report, Deliverable No.8, DSSCV, IST-2001-35443, 15. September 2003.

[43] K. Sundaraj and C. Laugier. Fast contact localisation of moving deformable polyhedras. In *IEEE Int. Conference on Control, Automation, Robotics and Vision*, Singapore (SG), December 2000.

[44] C. Tzafestas and P. Coiffet. Real-time collision detection using spherical octrees : Vr application, 1996.

[45] G. van den Bergen. Proximity queries and penetration depth computation on 3d game objects. *Game Developers Conference*, 2001.

[46] L.J. van Vliet, I.T. Young, and P.W. Verbeek. Recursive Gaussian Derivative Filters. In *Proceedings of the 14th International Conference on Pateern Recognition ICPR'98*, volume 1, pages 509–514., Brisbane, Australia, 1998. IEEE Computer Society Press.

[47] Pascal Volino and Nadia Magnenat Thalmann. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In Dimitri Terzopoulos and Daniel Thalmann, editors, *Computer Animation and Simulation '95*, pages 55–65. Springer-Verlag, 1995.

[48] Pascal Volino and Nadia Magnenat-Thalmann. *Virtual Clothing, Theory and Practice.* Springer-Verlag Berlin Heidelbarg, 2000.

[49] CMLabs Vortex. http://www.cm-labs.com/products/vortex/.

[50] J. Weickert, K. J. Zuiderveld, B. M. ter Haar Romeny, and W. J. Niessen. Parallel implementations of AOS schemes: A fast way of nonlinear diffusion filtering. In *Proceedings of the 4th International Conference on Image Processing*, volume 3, pages 396–399, Santa Barbara, CA, USA, October26–29 1997. IEEE Computer Society Press.

[51] A. P. Witkin. Scale–space filtering. In *Proc. 8th Int. Joint Conf. on Artificial Intelligence (IJCAI '83)*, volume 2, pages 1019–1022, Karlsruhe, Germany, August 1983.

[52] G. Zachmann. Rapid collision detection by dynamically aligned dop-trees, 1998.