Collision Prediction Using MKtrees

M. Franquesa-Niubó
Dept. LSI
UPC
Av. Diagonal 647
08028 Barcelona. Spain
marta@lsi.upc.es

P. Brunet
Dept. LSI
UPC
Ed. A0 Campus Nord
08028 Barcelona. Spain
pere@lsi.upc.es

ABSTRACT

In this paper, the collision prediction between polyhedra under screw motions and a static scene using a new *K* dimensional tree data structure (*Multiresolution Kdtree*, *MKtree*) is introduced. In a complex scene containing a high number of individual objects, the MKtree represents a hierarchical subdivision of the scene objects that guarantees a small space overlap between node regions. The proposed MKtree data structure succeeds in performing simultaneously space and scene subdivision. MKtrees are useful for *broad phase* collision and proximity detection tests and for time-critical rendering in large environments requiring external memory storage. The paper proposes an efficient broad phase collision prediction algorithm. Examples in ship design applications are presented and discussed..

Keywords

Collision Prediction and Detection, Large Environments, Hierarchical Representations, Virtual Reality, Screw motions.

1. INTRODUCTION

Our original interest is related to collision detection and prediction in ship design applications. The goal is to predict collisions between oil tanker objects and a moving object (with application to maintenance tasks). In ship design applications, the number and distribution of objects are complex. Thus, the number of objects involved is high and therefore the environment is considered *Large*.

To model the geometry of the oil tankers to speed up the collision test, and to well manage information between main memory and disk we use the *MKtrees*. MKtrees were aimed to answer in a efficiently way when a collection of objects can collide, when a new object introduced in the geometrical model collides with one existing object.

WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

Computing collision queries amongst moving many 3D objects and between moving objects and static obstacles has been a subject of research in several areas as: virtual reality, robotics, medical, etc. We address here on rigid polyhedra, and do not focus on collisions of deformable models.

When complex 3D motions are involved, the locations of each object can be evaluated using a series of small time increments. At each stage of this simulation, the transformed instance of each object is tested against the instances of other objects using a static interference test [Lin91]. Many techniques have been proposed to reduce the bottleneck of testing for interference between *all pairs* of *N* objects, $O(N^2)$.

Algorithms [Dob90, Sei90, Chu96, Lin91, Mir98, Gil88, Cam97] and models have been introduced, hierarchical models [Pob92, Bart96, Hub96, O'S99, Klo98, Zac02,Got00, He99, Fra03a] and non--hierarchical ones [Bor03, McN99].

The above approaches are based on *Multiple interference tests* making use of static interference tests sampling time. Redon et al. [Red02] proposed collisions prediction solution which also makes use of screw displacement, they used the OBB (object oriented bounding box) hierarchies.

Kim and Rossignacc [Kim03] proposed a collision prediction approach, in which they compute the

time and location of collisions directly from the relative motion of pairs of objects. They detect all occurrences of face/vertex and edge/edge collisions and report the first one to occur.

To obtain a simple formulation of the exact collision parameters, we approximate the relative motion (the relative motion of an object *A* with respect to a moving object *B* is the motion of *A* in the body coordinate system of *B*) between any two objects by a continuous series of screw motion segments by using the technique presented by Kim and Rossignac in [Kim03]. For each screw motion segment, their method computes the times of collision between all vertices of the first object and the faces of the other, and between each edge of the first object and each edge of the second object. They report the smallest of these times. The method assumes that objects are initially disjoint.

Our environment is composed by hundreds of thousands of polygons in a closed space, the usual case in ship design applications.

The set of objects that are belonging to each oil tanker constitute the static obstacles with which a moving object can potentially collide. As the number of static obstacles is high, their geometry can not fit in main memory at once. Then, to store the environment objects geometry and to handle them in an efficient way, we designed a hierarchical model based on out-of-core techniques named MKtrees [Fra03a].

Efficient handling and spatial searching in complex systems has been studied by several authors, also, obtaining solutions based on different data structures.

The first application that we have considered was the efficient access to objects in a data structure to solve collision and proximity detection between objects. In [Lin98, Jim01, Fra03b] a recent surveys can be found. Most of the algorithms use bounding volume hierarchies to approximate geometry of objects, such as OBBtrees (object oriented bounding box trees) [Got96, Bar96, Got00], sphere-trees [Pal95, Hub96, O'S99, Bra03], AABBtrees (axis aligned bounding box trees) [Coh95, Hel95, Ber97, Kle03] and *K*-dops (discrete oriented polytopes) [Hel96, Klo98, He99].

In this paper we combine the use of the hierarchical model *MKtree* with an algorithm based on the method proposed by Kim and Rossignac [Kim03] to compute collision prediction in the ship design applications environment.

In the next Section 2 we state the problem of collision detection in our environment. In Section 3, we present the collision detection algorithm. In Section 4, we provide the results of our implementation and the discussion of our method. Finally, in Section 5 the conclusions and future work are presented.

2. THE PROBLEM

The main problem that we are interested to solve is to predict the collision between a moving object *A* and objects that belong to a complex static environment (an oil tanker in our case). The oil tanker environment is modelled by using the MKtree. To represent the moving object *A* (that follows an screw motion trajectory) we store: The bounding sphere of *A*: *S(A)* and the polygonal representation (triangles): *P(A)*. To represent the set of static obstacles *B* we store their *MKtree* [Fra03c]. The MKtree:

- Is a hierarchical representation based at the same time in spatial and scene object partition.
- Spaces associated to different subtrees do not overlap.
- Each leaf of the MKtree is corresponding to one block on disk
- Each leaf of the MKtree stores:
    - The axis-aligned-bounding-box of each object, O_i, belonging to the leaf: AABB(O_i)
    - The set of bounding spheres that cover each object S(O_i)
    - The polygonal representation of each object P(O_i)

The MKtree generation algorithm for a set of objects S_n recursively computes the best dimension and location to divide the initial list S_n in the sense of minimizing the number of objects that overlap the intersection space between the bounding boxes of the two divided sublist S_{n1} and S_{n2}. As the overlap region is not always null [Fra03c], we split this region in order to achieve a complete object and space division.

Overlap splitting is represented by the so called *SplitTree*. The *SplitTree* generation algorithm is based on the hypothesis that, for any straight line *h* parallel to the splitting direction, a point *Z* belonging to *h* exists such that *Intersection*(S_{n1},*h*) is separated from *Intersection*(S_{n2}},*h*) by the point *Z*. We have observed that this hypothesis is fulfilled in our practical examples (spatially disjoint scene objects required).

In fact, the SplitTree is a kind of Kdtree that splits the overlap region by representing the set of splitting points Z [Fra03c]. The SplitTree is very compact, as every node is only storing the corresponding splitting half space. It is stored as part of the corresponding node n of the MKtree.

3. COLLISION PREDICTION

In this section, our proposed algorithm that consists of four steps is presented. The first two ones solve the collision prediction in the entire *broad phase* and the two last solve the collision in the entire *narrow phase* [Kit94, Hub95, O'S99, Fra03b]. The algorithms are based on elementary set of tests that are detailed below. The algorithm is based on:

- Test simplifications due to the screw motions features,
- The double partition criteria (space and scene objects) of the MKtree and,
- The MKtree representation on disk

The first step of our algorithm is a prediction test between the bounding sphere *S(A)* of the moving object *A* and the hierarchical MKtree representation of the oil tanker. This is the main part of the prediction test and, works by traversing the MKtree and detecting the leaves of the tree that collide with the sphere *S(A)* under screw motion.

The second step uses the multiresolution structure of our MKtree and refines the predicted time intervals for collision between *S(A)* and the tree leaves. It uses a bounding set of spheres approximation of the objects in individual tree leaves.

The last two steps of the main algorithm use directly the test from Kim and Rossignac [Kim03] and refine the predicted collision time intervals using the polygonal representation of the individual ship objects. The four steps are presented in the next subsections.

3.1 Prediction Test between S(A) and the MKtree(B)

The goal of this first test is to select the candidate leaves of the MKtree that can be involved in a collision with *A*. Only the leaf candidates will be retrieved from disk. This fact, will reduce the number of accesses to disk and will speed up the whole process of collision query.

The first call to this test is done with:

- The bounding sphere of *A*, *s*
- The screw motion of object *A*, *m*
- Initial time interval in which the collision can be found, *Int*. This is the time interval between the beginning and the final of the screw motion.
- The root node of the MKtree, *n*

{Prec: n is a node of the MKtree and Int is not void}

procedure SphereTree(**inp** s: sphere, m:motion, Int:TimeInt, n:node)

{Post: returns the list of potential leaves of the MKtree that can collide with the bounding sphere s of A. Every list element contains a pointer to a MKtree node leaf n and the time interval of potential collision between s and n }

procedure SphereTree(**inp** s: sphere, m: :motion, Int: TimeInt n: node)

{ Compute a sharp time interval }

t_{int} = SphereBox(s,m,n,box,Int)

if

IsLeaf(n) → AddToLeafList(n, t_{int})

Void(SphereBox(s,m,(n.sr).box,t_{int}) → SphereTree(s,m,t_{int},n.sl)

Void(SphereBox(s,m,(n.sl).box,t_{int})→ SphereTree(s,m,t_{int},n.sr)

otherwise → *{ the swept volume caps the overlap volume }*

b_{int} = Intersection(b, (n.sr).box, (n.sl).box)

t_{int} = SphereBox(s,m,b_{int},t_{int})

n_s = SplitTreeRootNode(n)

case

SphereSplitTree(s,m,b_{int},t_{int},n_s)=left→
SphereTree(s,m,t_{int},n.sl)

SphereSplitTree(s,m,b_{int},t_{int},n_s)=right→
SphereTree(s,m,t_{int},n.sr)

SphereSplitTree(s,m,b_{int},t_{int},n_s)=both →
SphereTree(s,m, t_{int},n.sl)
SphereTree(s,m, t_{int},n.sr)

endcase

endif

endprocedure

Where: *n.sr*, *n.sl* and *n.box* are the son right pointer, son left pointer and the bounding box of the node *n*, *AABB(n)*, respectively, b_{int} is the resulting intersection box between *(n.sr).box* and *(n.sl).box* and, finally, n_s is the root node of the *SplitTree* associated to a node *n* of the MKtree. All arguments to the previous procedure are input parameters.

The function *SphereBox* predicts the interval time of collision between a moving sphere and a box. It uses a conservative test and computes the intersection between a point (the center of the sphere) and an extended box (inflated box with the sphere radius).

Equations (9)-(13) from [Kim03] are applied to the six oriented planes of the extended box, and the corresponding intersection times (with an entering-

-existing flag) are sorted in order to obtain a first approximation of the predicted collision time interval. The time interval extremes are finally refined using a Newton-Raphson technique.

The *SphereSplitTree* function is called when *s* intersects the overlap zone of the node *n* and, it returns if the swept volume of *s* intersects with the left son, only, with the right son, only, or with both. The SplitTree is traversed and the sphere motion is classified with respect to the planes that split the overlap region between the two son nodes of *n*.

3.2 Prediction Test between S(A) and a leaf of the MKtree n

This test is performed for each leaf of the computed list in the *SphereTree* (previous subsection). For each leaf in the list, *S(A)* is tested against each sphere of the objects of the list:

{Prec: n is a leaf node of the MKtree, Int is not void}

function SphereLeaf(**inp** s:sphere, m:motion, Int: TimeInt, n: node) **return** TimeIntList

{Post: returns the list of potential collision time intervals between the bounding sphere s of the object A and the bounding spheres of the MKtree node n}

function SphereLeaf(**inp** s:sphere, m:motion, Int: TimeInt, n: node) **return** TimeIntList

InitializeTimeIntervalList(tL)

for each bounding sphere b in n **do**

t_{int}= SphereSphere(s,m,b,Int)

if no Void(t_{int}) → InsertAndMerge(t_{int}, tL) **endif**

enddo

return tL

endfunction

The *InitializeTimeIntervalList(tL)* initializes the *tL* list to the empty list. In the *InsertAndMerge* procedure, if the new time interval t_{int} caps with any of the list *tL*, then the union of both is what is inserted in the list. The geometric test *SphereSphere* is based on a point-sphere test (this sphere having a radius equal to the sum of the radii of the initial spheres) and uses an iterative algorithm similar to the *SphereBox* algorithm.

3.3 Prediction Test between P(A) and P(n)

We use this test for the leaves that have a non-null time interval after the test described in the previous Section. In this case, we directly use the algorithm [Kim03] between the polygon al model of *A* and,

- A bounding polyhedron [And01] of the candidate objects in the tree leaf. This test is used as a first filter and the test has a limited complexity order because bounding polyhedra of not more than hundred faces are used.
- The polyhedral model of the candidate objects in all the cases where the result of the test with the bounding polyhedron is non-null time interval

4. RESULTS

We have generated several MKtrees from several input data. All the data are portions of oil tankers.

Figures 1, 2, 3 and 4 show general and detailed views of two of the oil tankers.

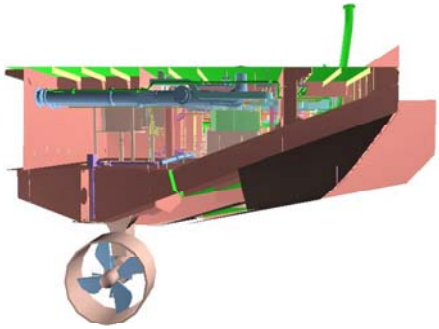

Figure 1. General view of one oil tanker

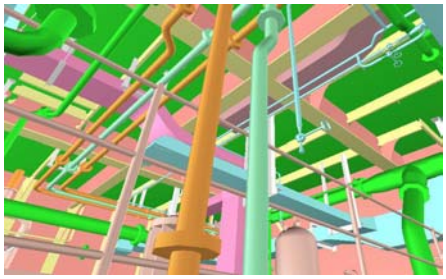

Figure 2. Inside view of one oil tanker

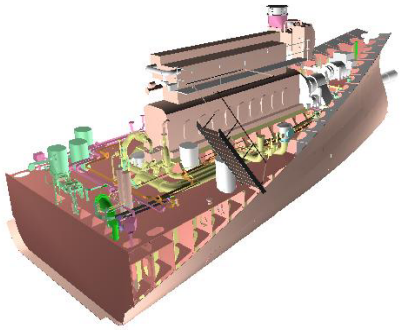

Figure 3. External view of one oil tanker

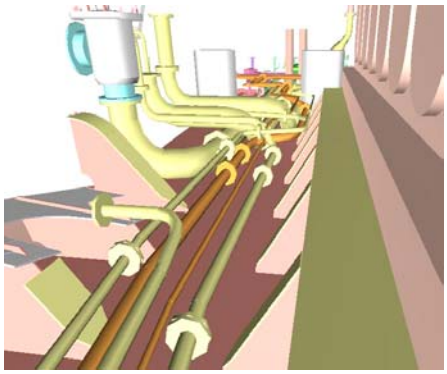

Figure 4. Detailed view of one oil tanker

Table 1 shows the input data description and the MKtree results. In the two first columns, the input data associated to the oil tankers are described and, in the rest of columns, the information of each MKtree generated is presented. Each column in the Table 1 means:

- $\mathbf{N_{pol}}$: Number of polygons of the corresponding oil tanker.
- $\mathbf{T_L}$: Maximum MKtree level.
- $\mathbf{N_g}$ and $\mathbf{N_L}$: Number of grey nodes and number of leaves in the MKtree, respectively.
- $\mathbf{SpT_L}$: Average SplitTree level.
- **Mem**: Memory occupancy of the MKtree.

Oil tanker	$\mathbf{N_{pol}}$	$\mathbf{T_L}$	$\mathbf{N_g}$	$\mathbf{N_L}$	$\mathbf{SpT_L}$	**Mem**
ALL	139460	17	656	657	2.2	29842
LOW2	131849	21	1284	1285	2.7	70937
MODM	156767	21	1290	1291	2.4	67334
SBL	131323	17	376	377	2.1	18696
SC32	181972	16	1430	1431	2.4	74641
SC33	167059	19	1133	1134	2.2	57204
SFACL	169128	21	1406	1407	2.6	76147
SFACM	161257	20	1420	1421	2.6	76905
TEX	106703	21	810	811	3.0	43742

Table 1. Results of MKtree generated from several oil tankers

On the other hand, in our present example the moving object *A* is composed by *500* polygons and it is bounded by a bounding sphere with a diameter of *200mm*.

As a consequence of the *MKtree* building procedure that generates the tree taking into account the double partition criteria: space and scene object partition, in the limit as object A decreases in size the time intervals will become disjoints and the algorithm of prediction detection gives the minimum number of leaves that can be involved in a collision.

There are four parameters that describe a trajectory under screw motion (see [Kim03] for more detailed information). These parameters define the direction of the screw axis **v**, a fixed point on this axis **p**, the translation speed *d* along **v**, and the speed of rotation *b* around the axis parallel to **v** and passing through **p**. Another argument value that we need to compute prediction is the point location of the center of the sphere *S(A)* relative to coordinate world of the MKtree at initial time, say **q(0)**. Then,

$$\mathbf{q(t)} = \mathbf{p} + t*d*\mathbf{v} + Rotation(\mathbf{q(0)} - \mathbf{p}, t*b)$$

Where boldface parameters indicate vectors and where the *Rotation* described a rotation of the vector $(\mathbf{q(0)} - \mathbf{p})$ by an angle $t*b$.

Several screw motions for the moving object *A* have been taken into account. Each motion is described by the above trajectory parameters with **v** = (1,0,0) and with the rest of values described in Table 2.

motion	*d*	**p**	*b*	**q(0)**
m_1	40	$\mathbf{p} = \mathbf{q}$	0	CR_x, CR_y, CR_z
m_2	0	$\mathbf{p} \neq \mathbf{q}$	20	$CR_x, CR_y + BR_y/10, CR_z$
m_3	0	$\mathbf{p} \neq \mathbf{q}$	20	$CR_x, CR_y + BR_y/3, CR_z$
m_4	0	$\mathbf{p} \neq \mathbf{q}$	20	$CR_x + BR_x/3, CR_y, CR_z$
m_5	40	$\mathbf{p} \neq \mathbf{q}$	20	$CR_x, CR_y + BR_y/10, CR_z$
m_6	10	$\mathbf{p} \neq \mathbf{q}$	20	$CR_x + BR_x/10, CR_y, CR_z$
m_7	10	$\mathbf{p} \neq \mathbf{q}$	20	$CR_x, CR_y, CR_z + BR_z/4$
m_8	10	$\mathbf{p} \neq \mathbf{q}$	30	$CR_x + BR_x/10, CR_y, CR_z$

Table 2. Screw motion input argument values

In Table 2 (CR_x, CR_y, CR_z) is the center coordinate of the *AABB(n)* box of the MKtree root node *n* and, BR_x, BR_y and BR_z are its dimensions.

The results of the collision prediction algorithm are presented in Table 3. Each cell number in Table 3 is the number of candidate leaves that potentially collide with the moving object *A*.

Ship	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
ALL	22	34	30	3	43	12	13	10
LOW2	10	11	3	8	2	22	42	32
MODM	28	50	13	15	13	70	60	60
SBL	1	2	2	1	2	2	3	1
SC32	19	37	8	13	6	63	43	59
SC33	4	13	21	0	58	11	1	11
SFACL	13	6	8	7	11	31	11	25
SFACM	22	39	5	18	5	69	42	54
TEX	23	16	1	9	0	20	21	17

Table 3. Results of Collision Prediction Alg1orithm applied to the MKtrees of Table 1.

Looking at this last Table we can observe and conclude that the number of candidate leaves involved in a possible collision with object *A* is low compared with the total number of the leaves of the MKtrees (from Table 1). For example, in the MKtree of the oil tanker SC32 with 1431 leaves (see Table 1), the number of candidate leaves involved in a collision with object *A* moving under a trajectory as m_3 is only 8.

Now, taking into account that each leaf on disk has a maximum size of 20 blocks (each one of 512 bytes), this means that for this oil tanker and motion only 160 blocks will be retrieved from disk versus 28620 of the original MKtree (corresponding to the 1431 leaves). This is true for all the screw motion trajectories and oil tankers (as it can be seen in tables 1 and 3).

Then, we can conclude that the use of MKtrees to compute collision prediction permits to reduce the number of disk access considerably.

In fact our algorithm not only determines the candidate leaves of the MKtree for collision but also computes a bound on the predicted collision time interval. For instance, in the case of motion m_2 and the model SFACL we have the results exposed in Table 4.

Extensive results of algorithms from Subsections 3.1 and 3.2 (that are not presented here due to page restrictions) can be found in [Fra03d].

Leaf	Initial Collision Time	Final Collision Time
1150	0.01	0.24
1156	0.05	0.29
1148	0.06	0.27
987	0.10	0.22
1338	0.40	0.44
967	1.46	1.65

Table 4. Time intervals of Collision Prediction between each candidate leaf of the MKtree of the oil tanker SFACL and S(A) under m_2 motion.

5. CONCLUSIONS

In this paper we have combined the advantages of collision prediction making use of MKtrees to model the static obstacles against objects that are moving under screw motions using a method based on the Kim and Rossignac algorithm [Kim03].

The use of the MKtrees reduces considerably the main memory occupancy, the number of disk accesses and the execution time.

In the future, we will continue with the analysis and simulation of the presented algorithm (specially in the narrow phase) and we will examine other possibilities of bounding polyhedra for the objects in the MKtree leaves.

6. ACKNOWLEDGMENTS

This work has been partially supported by the CICYT project TIC-98-0586-C03-01.

7. REFERENCES

[And01] Andujar, C. and Brunet, P. and Ayala D. Topology-reducing surface simplification using a discrete solid. ACM Transactions on Graphics. Vol 21. No. 2. pp 88-105. April. 2002.

[Bar96] Barequet, G. and Chazelle, B. and Guibas, L.J. and Mitchell. J.S.B. and Tal, A. BOXTREE: A Hierarchical Representation for Surfaces in 3D. EUROGRAPHICS Conf. Proc, vol 15, pp 387-396. Blackwell Publishers. August. 1996.

[Ber97] Bergen, G. Van Der. Efficient Collision Detection of complex deformable models using AABB trees. Journal of Graphic Tools, vol 2, No 4, pp 1-14. 1997

[Bor03] Borro, D. Colisiones en Estudio de Mantenibilidad con Restitución de Esfuerzos sobre Maquetas Digitales Masivas y Compactas. PhD Thesis. Universidad de Navarra. 2003.

[Bra03] Bradshaw, G. and O'Sullivan, C. Adaptative Medial-Axis Approximation for Sphere--Tree

Construction. ACM Transactions on Graphics. Vol 22. No 4. 2003.

[Cam97] Cameron, S.A. Enhancing GJK: Computing Minimum Penetration Distances Between Convex Polyhedra. Proc. of the Int. Conf. on Robotics and Automation. pp 3112-3117, 1997.

[Coh95] Cohen, J. and Lin, M. and Manocha, D. and Ponamgi, K. I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scaled Environments. Proc. of ACM Int. 3D Graphics Conference. Pp 189-196. 1995.

[Chu96] Chung, K. and Wang, W. Quick collision detection of polytopes in virtual environments. Proc. of ACM Symposium on Virtual Reality Software and Technology. 1996

[Dob90] Dobkin D.P. and Kirkpatrick, D. G. Determining the separation of preprocessed polyhedra A unified approach. In Proc. 17th Internat. Colloq. Automata Lang. Program. Lecture Notes in Computer Science. Springer-Verlag. Vol 443. pp 400-413. 1990

[Fra03a] Franquesa-Niubó, M. and Brunet, P. Collision Detection using Mktrees. Proc. CEIG 2003. pp 217-232. July, 2003.

[Fra03b] Franquesa-Niubó, M. and Brunet, P. Collision Queries: Models and Algorithms. Technical Report. Software Dept. LSI. U.P.C. 2003. Ref: LSI-03-45-R http://www.lsi.upc.es/dept/techreps/techreps.html

[Fra03c] Franquesa-Niubó, M. and Brunet, P. MKtree: Construction and Applications. Technical Report. Software Dept. LSI. U.P.C. 2003. Ref: LSI-03-40-R. http://www.lsi.upc.es/dept/techreps/techreps.html Submitted to the Computer Graphics Forum Journal as: MKtrees and their applications

[Fra03d] Franquesa-Niubó, M. and Brunet, P. Collision Prediction using MKtrees: Broad Phase and Refinement Levels of the Narrow Phase. Technical Report. Software Dept. LSI. U.P.C. 2003. Ref: LSI-03-46-R http://www.lsi.upc.es/dept/techreps/techreps.html

[Gil88] Gilbert, E.G. and Johnson, D.W. and Keerthi, S.S. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal of Robotics and Automation. vol 4. No 2. pp 193-203. April, 1988.

[Got96] Gottschalk, S. and Lin, M.C. and Manocha, D. OBBtree: A Hierarchical Structure for Rapid Interference Detection. ACM SIGGRAPH Conf. Proc. pp 171-180. August, 1996.

[Got00] Gottschalk, S. Collision Queries using Oriented Bounding Boxes. PhD Thesis. University of North Carolina. Department of Computer Science. 2000.

[He99] He, T. Fast Collision Detection Using QuOSPO Trees. Symp. on Interactive 3D Graphics, Atlanta, ACM. pp 55-62. 1999.

[Hel95] Held, M. and Klosowski, J.T. and Mitchell, J.S.B. Speed Comparison of Generalized Bounding Box Hierarchies. Technical Report. Applied Math, SUNY Stony Brook. 1995

[Hel96] Held, M. and Klosowski, J.T. and Mitchell, J.S.B. and Sowizral, H. and Zikan, K. Real-Time Collision Detection for Motion Simulation within Complex Environments. Technical Report. Applied Math, SUNY Stony Brook. 1996

[Hub95] Hubbard, P. M. Collision Detection for Interactive Graphics Applications. IEEE Transactions on Visualization and Computer Graphics. Vol 1. No 3. pp 218-230. September, 1995.

[Hub96] Hubbard, P. M. Aproximating Polyhedra with Spheres for Time-Critical Collision Detection. ACM Transactions on Graphics. vol 15. No 3. pp 179-210. July, 1996.

[Jim01] Jimenez, P. and Thomas, F. and Torras, C. 3D Collision Detection: A Survey. Computers and Graphics. Vol 25. Num 2. pp 269-285. August. 2001.

[Kim03] Kim, B. and Rossignac, J. Collision Prediction for Polyhedra under Screw Motions. Proc. of the ACM SM'03. pp 4-10. Seatle, Washington. ISBN:1-58113-706-0. June, 2003.

[Kit94] Kitamura, Y. and Takemura, H. and Ahuja, N. and Kishino, F. Efficient Collision Detection Among Objects in Arbitrary Motion Using Multiple Shape Representation. Proceedings 12th IARP Inter. Conference on Pattern Recognition. pp 390-396. October, 1994.

[Kle03] Klein, J.and Zachmann, G. Probability-Guided Collision Detection. Technical Report. University of Paderborn. Ref: tr-ri-03-242.

2003. http://www.upb.de/cs/ag-madh/WWW/english/janklein.

[Klo98] Klosowski, J. T. and Held, M. and Mitchel, J. S.B. and Sowizral, H. and Zikan, K. Eficient Collision Detection Using Bounding Volume Hierarchies of K-DOPs. IEEE Transactions on Visualization and Computer Graphics. vol 4. No 1. pp 21-36. January-March. 1998.

[Lin91] Lin, M.C. and J.F. Canny . A Fast Algorithm for incremental distance calculation. Proc. of the IEEE Inter. Conf. on Robot and Automation, Sacramento. CA. pp. 1008-1014, vol 2. 1991.

[Lin98]Lin, M.C. and Gottschalk, S. Collision detection between geometric models: a survey. Proc. of IMA Conference on Mathematics of Surfaces. 1998.

[McN99] McNeely, W.A. and Puterbaugh, K.D. and Troy, J.J. Six-degrees-of-fredom haptic rendering using voxel sampling. Proceedings of SIGGRAPH 99. pp 401-408. ISBN: 0-20148-560-5. Los Angeles. CA. August, 1999.

[Mir98] Mirtich, B. V-Clip: Fast and Robust Polyhedral Collision Detection}. ACM Transactions on Graphics. vol 17. No 3. pp 177-208. July, 1998

[O'S99] O'Sullivan, C. and Dingliana, J. Real-time collision detection and response using sphefe-trees. In 15th Spring Conference on Computer Graphics. pp 83-92. ISBN: 80-223-1357-2. April, 1999. http://citiseer.nj.nec.com/301042.html

[Pal95] Palmer, I.J. and Grimsdale, R.L. Collision Detection for Animation using Sphere-trees. Computer Graphics Forum. 1995

[Pob92] Pobil, A.P. Del and Serna, M.A. and Llovet, J. A new representation for collision avoidance and detection. IEEE Int. Conf. on Robotics and Automation. vol 1, pp 246-251. Nice, France. May, 1992.

[Red02] Redon, S. and Kheddar, A. and Coquillart, A. Fast Continuous Collision Detection between Rigid Bodies. Proc. Eurographics. Ed. G. Drettakis and H.P. Seidel. Vol 21(3). September, 2002.

[Sei90] Seidel, R. Linear programming and convex hulls made easy. In Proc. 6th Ann. ACM Conf. on Computational Geometry. Berkeley, CA. pp 211-215. 1990

[Zac02] Zachmann, G. Minimal Hierarchical Collision Detection. Proc. ACM Symposium on Virtual Reality Sotfware and Technology (VRST). pp 121-128. ISBN:1-58113-530-0. Hong Kong, China. November. 2002