EXPLOITING ADVANCED COLLISION DETECTION LIBRARIES IN A PROBABILISTIC MOTION PLANNER

S. Caselli, M. Reggiani, M. Mazzoli

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma,
Parco Area delle Scienze, 181A,
43100 Parma,ITALY
{caselli, reggiani, mazzoli}@ce.unipr.it

ABSTRACT

Motion planning is a fundamental problem in a number of application areas, including robotics, automation, and virtual reality. The performance of motion planning is largely affected by the underlying collision detection technique. In this paper we report the results of an experimental evaluation of several recent collision detection libraries within the context of motion planning for rigid and articulated robots in 3D workspaces. The libraries investigated have been chosen based also on their free availability to the research community. Results reported in this paper show that some of the collision detection packages investigated are very sensitive to the type of problem to be solved, possibly determining the best performance on certain problems and proving very inefficient or even not applicable on different problems. Other collision detection libraries are much less sensitive to the type of problem, although they do not necessarily exhibit the best performance on any given problem. These considerations suggest that a motion planner could take advantage from the ability to select one among a range of collision detection libraries based on characteristics of the problem to be solved which could be known a priori.

Keywords: motion planning, collision detection

1 INTRODUCTION

Motion planning is a fundamental problem in a number of application areas, including robotics, automation, and virtual reality. Given an initial and a final configurations, the goal of the basic motion planning problem is to find a path starting at the initial configuration and terminating at the goal configuration, while avoiding collision with the obstacles [Gupta96]. In an application area whose goal is to find a collision-free path, the overall execution time is greatly affected by the quality of the collision detection algorithm adopted by the planner. In *complete planners*, execution time is dramatically influenced by the efficiency of collision detection. In these planners, indeed, the whole connectivity of configuration space (C-space) must be constructed, requiring, for every robot configuration, a collision check against all the obstacles in the environment. In *probabilistic planners* the C-space is incrementally explored, therefore only a subset of robot configurations need to be checked for collision against obstacles. Nevertheless, in most randomized path planner a large fraction of the overall execution time is spent for collision detection.

We are currently developing a new path planning tool called *parallel Potential Field Planner (pPFP)*. pPFP is a probabilistic planner based on a random heuristic search of the path guided by a potential function and aims to address motion planning of complex systems. pPFP builds upon the approach pioneered in [Barra91, Latom91]. The goal of our tool is to overcome some of the limitations of the early potential field approach [Barra91] and thus expand its range of applicability, while retaining its most valuable characteristics such as ease of use and probabilistic completeness. Namely, pPFP improves planning efficiency in escaping from local potential field minima by supplementing the random mo-

Library	Research Group	Available at:
Rapid	Gamma Research Group (NC at Chapel Hill)	`http://www.cs.unc.edu/~geom/OBB/OBBT.html`
V-Clip	MERL - Mitsubishi Electric Research Lab.	`http://www.merl.com/projects/vclip/`
SOLID	Comp. Graph. Group (Eindhoven Univ.)	`http://www.win.tue.nl/~gino/solid/`
V-Collide	Gamma Research Group (NC at Chapel Hill)	`http://www.cs.unc.edu/~geom/V_COLLIDE/`
PQP	Gamma Research Group (NC at Chapel Hill)	`http://www.cs.unc.edu/~geom/SSV/`

Table 1: Collision Detection Algorithms investigated.

tion strategy with more informed search strategies [Casel01]. Furthermore, it integrates multiple search strategies of the potential field planner into a parallel computation scheme which proves highly effective for many degree of freedom (d.o.f.) problems [Casel02]. The current implementation of the tool also includes exploitation of past experience [Casel00]. A recent assessment [Casel02] has shown that pPFP does provide good performance on the case studies investigated so far, and we suggest that it should be considered *au pair* with advanced path planners based on alternative approaches [Gupta98]. pPFP performance largely relies on a careful choice of a suitable collision detection library among the packages freely available to the research community.

This paper reports the experimental evaluation that supported our choice. We have restricted our analysis to the context of motion planning for rigid and articulated robots in 3D workspace. The narrowness of this analysis allows the design of tests able to measure robustness and effectiveness of the packages investigated in the solution of planning problems with different characteristics.

The paper is organized as follows. Section 2 provides a brief survey about the characteristics of evaluated libraries for collision detection whose source is free for non-commercial use. Section 3 describes the details of chosen testbeds, the experimental methodology adopted, and the performance of evaluated collision detection algorithms when integrated in our planner. A final section summarizes the contributions of the paper.

2 EVALUATED COLLISION DETECTION LIBRARIES

Our investigation has considered five collision detection libraries. The selection was based on their free availability for non-commercial use and on their ability to answer to the simplest query, i.e. whether two models touch. The libraries use polygonal models, either relying on polygon soup or on convex objects or objects composed of convex parts.

V-Clip [Mirti98] The Voronoi Clip, or V-Clip is a feature-based algorithm derived from the Lin-Canny algorithm [Lin91]. Its key notion is the Voronoi region [Meyer86] associated with every feature (vertex, edge, or face) of a polyhedron. Using Voronoi regions, the Lin-Canny algorithm tracks the closest features between a pair of convex polyhedra. Once these features are known, the closest points between them, and therefore between the polyhedra, can be determined. The V-Clip algorithm improves Lin-Canny thanks to its ability to handle penetration cases (a situation where Lin-Canny's termination criteria are never satisfied and the algorithm cycles forever) and its robustness (Lin-Canny can cycle forever in presence of geometric degeneracies). V-Clip exploits routines in the Qhull library [Barbe96] to build hierarchies of convex components from vertex sets. When the convex decomposition has a moderate number of parts, and the hierarchy is only a few levels deep, V-Clip can work well even with nonconvex objects.

The collision detection libraries coming next exploit a hierarchy of bounding volumes to reduce the number of primitives that need to be checked for contact. Different bounding volumes have been proposed in an effort to find the optimal decomposition for the conflicting goals of tight approximation of the input objects and rapid intersection test computation.

RAPID [Gotts96] The RAPID library is based on two algorithms. The first one uses a top-down decomposition technique that builds a hierarchy of Oriented Bounding Boxes (OBBs) of an input polygon soup model. An OBB is a bounding box whose orientation is arbitrary and the resulting hierarchy is called OBBTree. The second algorithm is exploited for collision tests among OBB pairs. The main idea is to start to verify whether two high level OBBs overlap. If they do not overlap, then the two models do not collide and the algorithm ends. Otherwise the algorithm verifies the overlapping of lower level OBBs. It has been shown that OBB overlap detection requires only fifteen simple axis projection tests (*separating axis theorem* [Gotts96]). OBBs fit the object tighter than axis-aligned boxes or spheres. This

results in a less deep hierarchy and, therefore, in better performance, since a lower number of overlapping tests is performed.

SOLID [van d99b] The SOLID library uses two algorithms. The first one builds a bounding volume hierarchy composed of Axis-Aligned Bounding Boxes (AABB). In [van d97], van den Bergen presents the decomposition algorithm showing a way to speed up overlap tests between AABBs and thus aligning AABB to OBB performance for rigid models. For deformable models, AABB are even faster to build and to update. The SOLID library also exploits an enhanced version of the Gilbert, Johnson and Keerthi (GJK) algorithm [Gilbe88], which computes distance between two convex polytopes using Minkowski difference and convex optimization techniques. The SOLID improvement over earlier GJK [Gilbe88, Camer97] is discussed in [van d99a], along with the obtained performance, robustness, and versatility. Finally, for convex hull computation SOLID relies on the Qhull library.

PQP [Larse99, Larse00] PQP employs a top-down strategy to create a hierarchy of Rectangle Swept Spheres (RSS) from a polygon soup model. A RSS is a volume covered by a sphere whose center is swept over a 3D rectangle. Performance analysis comparing RSS and other type of bounding volumes can be found in [Larse99] Once the RSS tree is available, distance computation between RSSs are performed. Instead of existing algorithms for distance computation among convex shapes [Gilbe88, Lin91], a specialized algorithm [Larse00] is used in order to improve efficiency (decreasing the number of distance computation operations), and robustness (ability to cope with numerical errors and degeneracies).

V-Collide [Hudso97] V-Collide unifies two framework: RAPID and I-Collide [Cohen95], a library for collision detection whose core is the original Lin-Canny algorithm [Lin91]. First, the n-body sweep-and-prune algorithm of I-Collide is used to filter collisions among a large number of objects to determine potential overlapping pairs. In the second stage, a pairwise test from the RAPID library determines whether the objects received from the first stage actually collide.

3 EXPERIMENTAL EVALUATION

In this section, we describe the details of chosen testbeds, the adopted experimental methodology, and the performance of evaluated collision detection algorithms when integrated in our planner.

3.1 EXPERIMENTAL SETTING

A first problem in the comparison of collision detection algorithms for probabilistic motion planners is the large number of forces influencing their performance. The complexity of the workspace description, the type and position of the obstacles, the average distance among obstacles and robot can greatly modify the final execution time.

In the effort to cover different environments, we designed several artificial problems involving obstacles with different characteristics (Fig. 1(a)-1(c)), and we studied also a complex CAD-type environment with a rigid robot (Fig. 1(d)) representing more realistic motion planning problem. The CAD model was designed by Boris Yamrom of the Computer Graphics & Systems Group at GE CRD and is available from the Robotics Laboratory of Texas A&M University (`http://www.cs.tamu.edu/faculty/amato/dsmft/benchmarks/`).

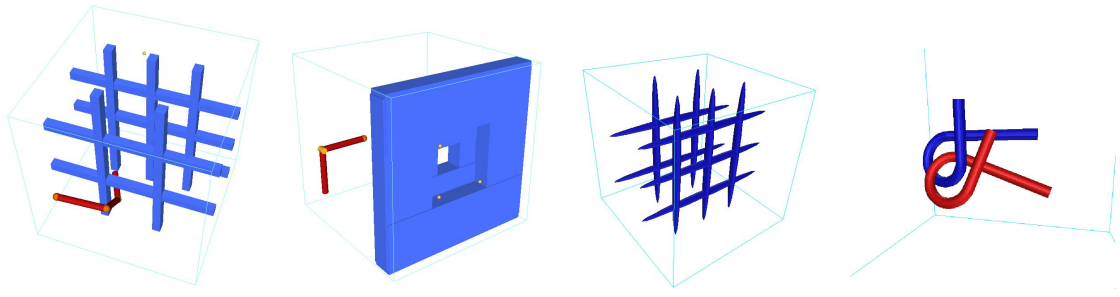
Another issue of collecting experimental results about probabilistic planners is the high variance in both computation time and quality of the solution found (related to the path length) due to their randomness. To account for this problem, instances with fixed random seeds have been studied for each problem. Due to the robustness of the evaluated collision detection algorithms, the planner returns the same path for each instance regardless of integrated algorithm. Therefore the difference in solution time for each presented problem depends only on collision detection algorithm.

All results reported in this section have been obtained with our planner [Casel00, Casel01] on a Pentium III Xeon 550 MHz PC with 512MB main memory.

3.2 EXPERIMENTAL RESULTS

Fig. 2 shows execution times required to solve ten different instances of grid benchmark (Fig. 1(a)).

These instances are solved faster when the planner uses V-Clip library. Answers to collision detection queries benefit of the use of this feature-based algorithm and are therefore remarkable quicker than using libraries exploiting hierarchies of bounding volumes. This is an expected result because usually this class of algorithms is faster, and therefore preferable, when the models are well-behaved, of moderate size, and not exceedingly nonconvex [Mirti98]. Regarding the other



(a) (b) (c) (d)

Figure 1: Artificial environments: (a) grid benchmark: the planner is required to find a path for the 3-link, 11 d.o.f. robot when its movements are restricted in a narrow space among close obstacles (b) hole benchmark: to reach the goal the 2-link, 7 d.o.f. robot is required to enter the narrow hole (c) modified grid benchmark: a 8,900 triangles representation. CAD model (d) α puzzle: the objective is to separate the intertwined tubes.

libraries, RAPID and V-Collide exhibit close average times. The "sweep-and-prune" stage of V-Collide is usually an overhead and it is not efficient enough to reduce the RAPID solution times in this class of problems. Fig. 3 shows average execution time to check a single configuration in a set of instances of the grid and hole benchmarks. Collision checking for the second problem is faster on average. The different number of links for the robot involved in the benchmarks only partially justifies the lower execution time. The workspace of the first problem is crammed with a grid obstacle, this cause a large hit ratio (greater than 20% in our tests). Moreover the robot and obstacles are usually close, forcing the algorithms to descend the bounding volume hierarchies to check the configuration for collisions. In the second problem, instead, a single large obstacle fills the right area of the workspace. With this obstacle configuration the hit ratios are lower and the collision detection operations are faster. To confirm previous experimental results, we separated hitting and safe configurations. The resulting sequences lose temporal coherence but allow us to separate the study of time required for checking of hitting and safe configurations.

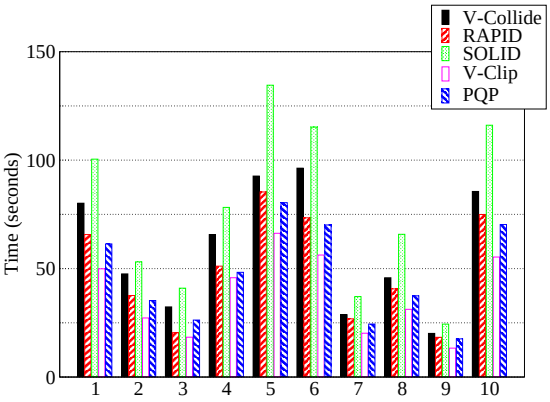


Figure 2: Execution time to solve ten instances of grid benchmark (Fig. 1(a)) with different collision detection libraries.

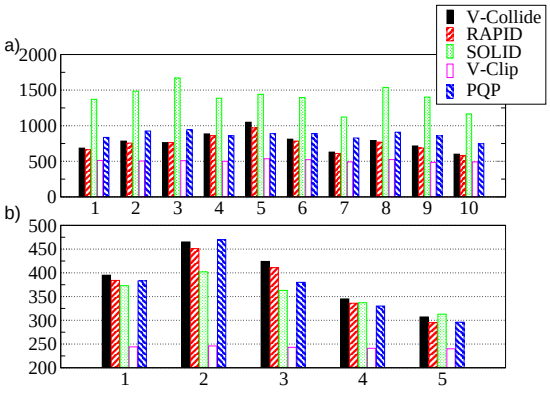


Figure 3: Average execution time to check a single configuration for a) grid (Fig. 1(a); 10 sequences) and b) hole (Fig. 1(b); 5 sequences).

As shown in Fig. 4, time required to check configurations that hit again obstacles are quite similar for the two benchmarks. Safe configuration checking is, instead, more expensive than hitting configuration checking for problem 1(a), the more obstacle crowded problem, and less expensive for the hole problem, due to its large open area. We found more difficult to explain why SOLID performs so satisfactory in the hole benchmark. A possible reason is that AABBs are particularly suitable for the obstacle characteristics of this

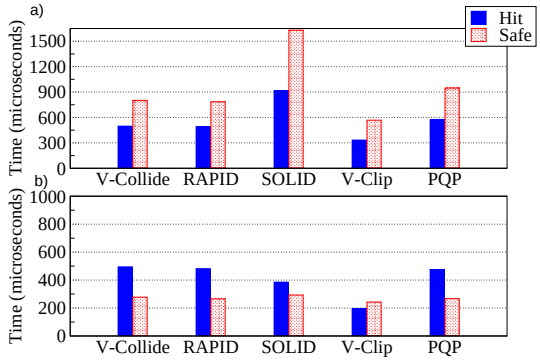


Figure 4: Average execution time (microseconds) required to check hitting and safe configurations for a) grid (Fig. 1(a)) and b) hole (Fig. 1(b)).

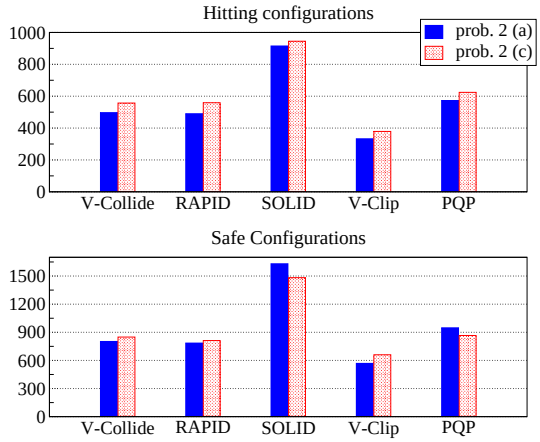


Figure 5: Influence of environment complexity on execution time (microseconds). Collision checking for problem 1(c) (8,900 triangles) is more expensive than problem 1(a) (100 triangles).

benchmark.

To consider the influence of environment complexity on execution time we modified the grid benchmark replacing the parallelepipeds with the inscribed ellipsoids (Fig. 1(c)), requiring 8,900 triangles for their description. As shown in Fig. 5, average execution times for both hitting and safe configurations increase over grid benchmark with parallelepipeds.

Last test is a complex CAD-type environment with rigid robots (Fig. 1(d)) consisting of two tubes, each twisted into an alpha shape; one tube is the obstacle and the other is the moving object (robot). The problem representation requires 1,008 triangles for each tube and the hit ratio of the sequence of configurations checked to find the final path is high (ranging from 30 to 40% of checked configurations in a set of ten solutions).

V-Collide	RAPID	SOLID
56.30	56.44	299.19
42.33	42.45	157.53
190.43	192.17	1007.35
15.46	15.76	33.15
29.26	29.38	88.73
165.32	166.22	967.11
86.40	87.24	364.21
29.66	29.16	84.18
25.48	25.34	132.27
46.49	46.52	251.64

Table 2: α puzzle execution time (ten different random seeds).

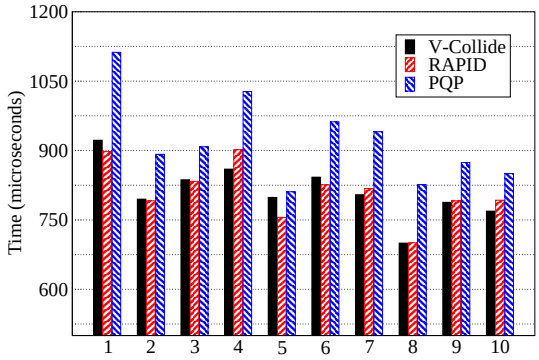


Figure 6: Average execution time to check a single configuration for the α puzzle with different collision detection libraries.

Feature-based or simplex-based algorithms such as Lin-Canny [Lin91] and GJK [Gilbe88], usually perform poorly when models are complex and non-convex. As shown in Table. 2, the planner with Solid is, indeed, two to six time slower than the planner with V-Collide or RAPID for this α puzzle.

Once again (Fig. 6),V-Collide and RAPID performances are comparable, while PQP suffers the high value of the hit ratio.

According to experimental results, V-Clip, a feature-based algorithm, is always faster and therefore should always be chosen when the models are well-behaved, of moderate size, and not exceedingly non-convex [Mirti98]. When the environment does not suit V-Clip requirements, either V-Collide or RAPID should be exploited for robot configuration checking. PQP execution times are close to V-Collide and RAPID ones but its performance seems negatively influenced by high hit ratios. Finally, SOLID should be used only with problems with narrow passages and space among obstacles large compared to robot dimensions.

4 CONCLUSIONS

In motion planning the overall execution time to find a collision-free path is greatly affected by the quality of the collision detection algorithm adopted by the planner. Results reported in this paper show that some of the collision detection packages investigated are very sensitive to the type of problem to be solved, possibly determining the best performance on certain problems and proving very inefficient or even not applicable on different problems. Therefore, the user should choose carefully the collision detection library when the characteristics of the problem are known a priori. Nevertheless, despite the number of survey on collision detection algorithms available from the literature [Lin98, Jimen, Jimen01], none of these works presents a speed comparison of the different libraries. Only partial experimental results can be found in [Mirti98], comparing speeds of *Lin-Canny*, *Enhanced GJK* e *V-Clip* algorithms and in [Larse99], comparing efficacy of different bounding volume trees.

The main contribution of this paper is a carefully experimental evaluation of collision detection packages for probabilistic motion planners. The narrowness of this analysis allowed the design of tests able to measure robustness and effectiveness of the investigated packages. In the near future we plan to study SWIFT++ [Ehman01], a new library recently released by the Gamma Research Group, whose performance are quite promising.

ACKNOWLEDGMENTS

We would like to thank Cristian Marastoni for the assistance with the integration and evaluation of PQP library. We would also thank the Research Groups that made available their collision detection software to the research community and prof. Nancy Amato (Texas A&M University) for the alpha puzzle benchmark.

Work on this paper has been supported in part by MURST, Italy (Project ISIDE, *Sviluppo di sistemi di elaborazione reattivi ed affidabili per applicazioni industriali*) and by a research project supported by ENEA on 'parallel visualization techniques for robot systems'.

REFERENCES

[Barbe96] B. C. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[Barra91] J. Barraquand and J.-C. Latombe. Robot motion planning: a distributed representation approach. *Internation Journal of Robotics Research*, 10(6):628–649, 1991.

[Camer97] S. Cameron. Enhancing GJK: computing minimum and penetration distance between convex polyhedra. In *IEEE International Conference on Robotics and Automation*, Albuquerque, NM, 1997.

[Casel00] S. Caselli and M. Reggiani. ERPP: an Experience-based Randomized Path Planner. In *IEEE International Conference on Robotics and Automation*, S. Francisco, CA, 2000.

[Casel01] S. Caselli, M. Reggiani, and R. Rocchi. Heuristic methods for randomized path planning in potential fields. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Banff, Alberta, Canada, 2001.

[Casel02] S. Caselli, M. Reggiani, and R. Sbravati. Parallel Path Planning with Multiple Evasion Strategies. submitted to the IEEE International Conference on Robotics and Automation, 2002.

[Cohen95] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-Collide: an interactive and exact collision detection system for large-scale environments. *ACM Int. 3D Graphics Conference*, 1, 1995.

[Ehman01] S. A. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Eurographics 2001*, Manchester, UK , 2001.

[Gilbe88] 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(2):193–203, 1988.

[Gotts96] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interface detection. In *ACM Siggraph*, 1996.

[Gupta96] K. Gupta. Pratical motion planning: An overview and state of the art. In *Workshop on Pratical Motion Planning in Robotics: Current Approaches and Future Directions*, Minneapolis, MN, 1996.

[Gupta98] K. Gupta and A. P. del Pobil. *Pratical Motion Planning in Robotics: Current Approaches and Future Directions*. Addison Wesley, 1998.

[Hudso97] T. Hudson, M. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-COLLIDE: accelerated collision detection for VRML. In *2nd Annual Symposium on the Virtual Reality Modelling Language*, Monterey, CA, 1997.

[Jimen] P. Jimenéz, F. Thomas, and C. Torras. Collision detection algorithms for motion planning. In J.-P. Laumond, editor, *Robot motion planning and control*, number 229, chapter 6. Lecture Notes in Control and Information Sciences.

[Jimen01] P. Jimenéz, F. Thomas, and C. Torras. 3D collision detection: a survey. *Computers and Graphics*, 25(2):269–285, 2001.

[Larse99] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, Chapel Hill, 1999.

[Larse00] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast distance queries with rectangular swept sphere volumes. In *IEEE International Conference on Robotics and Automation*, San Francisco (CA), 2000.

[Latom91] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Press, 1991.

[Lin91] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation*, Sacramento (CA), 1991.

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

[Meyer86] W. Meyer. Distance between boxes: applications to collision detection and clipping. In *IEEE International Conference on Robotics and Automation*, San Francisco (CA), 1986.

[Mirti98] B. Mirtich. VClip: fast and robust polyhedral collision detection. *ACM Transaction on Graphics*, 17(3):177–208, 1998.

[van d97] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphic Tools*, 4(2):1–13, 1997.

[van d99a] G. van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 2(4):7–25, 1999.

[van d99b] G. van den Bergen. *User's guide to the SOLID interference detection library*, 1999.