Continuous Collision Detection for Adaptive Simulations of Articulated Bodies

Sujeong Kim[1] Stephane Redon[2] Young J. Kim[1]
[1] **Ewha Womans University** [2] **INRIA**

Abstract

We present a novel algorithm to perform continuous collision detection (CCD) for articulated bodies when the motion of bodies is governed by an adaptive dynamics simulation. The algorithm is based on a novel hierarchical set of transforms to represent the kinematics of an articulated body recursively described by an assembly tree. The performance of our CCD algorithm significantly improves as the number of active degrees of freedom in the adaptive simulation of articulated bodies decreases.
Keywords: continuous collision detection, articulated body dynamics, adaptive dynamics, interval arithmetic.

1 Introduction

Collision detection (CD) is a problem of testing possible interference between geometric models in space. Many applications in different areas such as computer graphics, robotics and geometric modelling require fast and reliable CD to simulate the physical presence of virtual objects. As a result, CD has been extensively studied in the literature and many efficient CD algorithms are known. At a broad level, depending on how CD algorithms handle the motion of objects, they can be categorized into two types, *discrete CD and continuous CD*. Discrete CD algorithms check for interferences between static objects. Continuous CD (CCD) algorithms take the object's continuous motion into account and report the first time of contact (TOC) between moving objects if there occurs a collision. In the literature, there are six different approaches known for solving CCD for a single rigid body: algebraic equation solving approach [6, 7, 12, 18], swept volume approach [1], adaptive bisection approach [19, 26], kinetic data structures (KDS) approach [2, 13, 14], Minkowski sum-based approach [27], and conservative advancement [28]. CCD for articulated models have been proposed in [21, 22].

Recently, CCD has drawn much attention from different communities because of the need for correctly dealing with dynamic nature in applications. The major strength of using CCD in dynamic applications lies in a fact that the result of CCD always guarantees non-penetration between moving objects. For example, in rigid body dynamics, CCD has been used to enforce non-penetration constraints among simulated bodies [4, 20]. In constraint-based 6DOF haptic rendering, CCD can be used to compute the *God-object* of haptic probe that should not inter-penetrate the objects that the user is touching [17]. In sampling-based robot motion planning, it is crucial to find a continuous, collision-free path between two configurations of a moving robot, and CCD plays an important role in finding one [26, 25].

The use of forward dynamics has been considered as an effective way to control or simulate a large body of articulated characters in computer graphics and robotics [3, 5, 8, 11, 15, 9, 10]. However, it is becoming quite costly to simulate a complex scene with many articulated characters, with many degrees of freedom. Recently, a method has been proposed to rigorously simplify and speed up the computation of articulated-body dynamics, by predicting and simulating only a relevant subset of joints in the articulated bodies [23].

In this paper, we present a method to perform CCD for articulated bodies whose motion is governed by such an adaptive simulation. We demonstrate how a new hierarchical set of transforms can describe the kinematics of an articulated body. In particular, we show how this hierarchical set of transforms can be selectively and recursively updated during an adaptive articulated-body simulation. This enables us to roughly match the complexity of continuous collision detection to the one of the dynamics simulation, resulting in potentially significant speed-ups as the articulated-body dynamics are being simplified.

[1]Dept. of Computer Science and Engineering, Ewha Womans University, Seoul, Korea, Email: *kimsujeong@ewhain.net, kimy@ewha.ac.kr*
[2]INRIA Rhone-Alpes, France, Email: *stephane.redon@inria.fr*

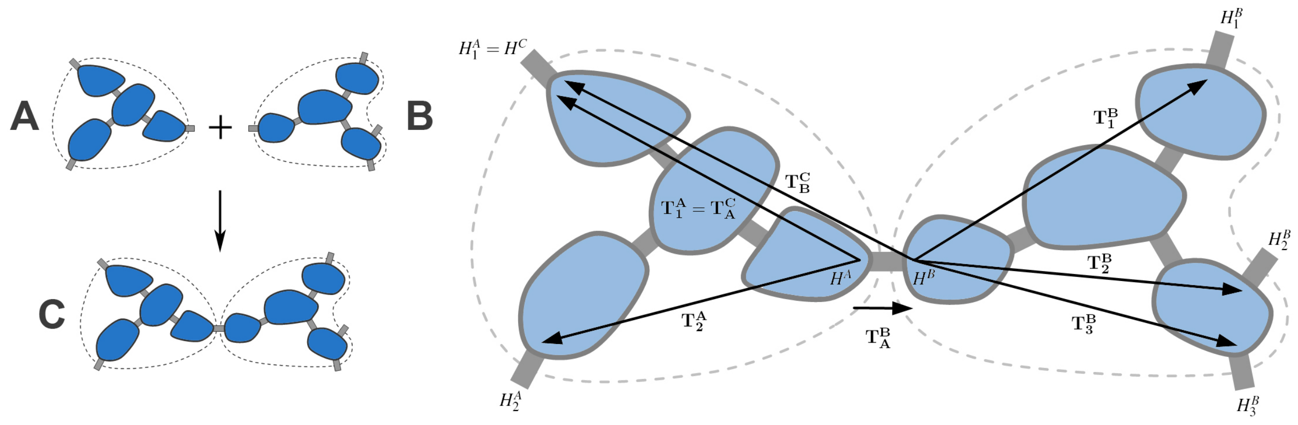

Figure 1: **The kinematic representation used in our continuous collision detection algorithm.** *The figure shows the principal joint transformation $\mathbf{T_A^B}$, the principal-to-secondary handle transformations $\mathbf{T_1^A}$, $\mathbf{T_2^A}$, $\mathbf{T_1^B}$, $\mathbf{T_2^B}$ and $\mathbf{T_3^B}$, and the child-to-parent transformations $\mathbf{T_A^C}$ and $\mathbf{T_B^C}$ (*cf *Section 2).*

2 Kinematics

In this section, we introduce a novel recursive representation of the kinematics of an acyclic branched mechanism. We start from a recursive definition of a branched articulated body proposed by Featherstone [9, 10], and introduce a set of transformations to describe the kinematics of the mechanism. Compared with the hierarchical representation we introduced earlier in [24], this novel representation is simpler and more efficient. In particular, updating the transforms of a node of the assembly tree now has a linear complexity in the number of handles of the node, compared to the quadratic complexity in the earlier work due to the quadratic number of transformations per node.

2.1 Definitions

As proposed in [9, 10], a (possibly branched) articulated body C is recursively defined as *two* articulated bodies A and B connected with a joint J, and the sequence of assembly operations is described in a binary *assembly tree*. In this recursive description, the leaf nodes of the assembly tree are rigid bodies, while the internal nodes represent sub-articulated bodies and the root node corresponds to the complete articulated body[1]. Equivalently, all internal nodes represent the joint used to perform the binary assembly operation.

In this representation, each sub-articulated body has a set of *handles*, *i.e.* locations where other sub-articulated bodies may be attached. For the sake of convenience, we call the handle H^A used to connect A to another articulated body the *principal handle* of A, while the k other free handles H_i^A of A $(1 \leqslant i \leqslant k)$ are called its *secondary handles*. Finally, if an articulated body C is formed by assembling A and B, we call the joint used to carry out the assembly the *principal joint* of C.

[1]Note how this differs from the traditional representation, in which a linkage is recursively defined by connecting *a single link* to a linkage.

To describe the kinematics of the mechanism, we rigidly attach a reference frame to each handle H, and define the following sets of rigid transformations:

1. **principal joint transformations**: the principal joint of each sub-assembly C with children A and B has an associated transformation $\mathbf{T_A^B}$, from (the reference frame of) the principal handle of A to (the reference frame of) the principal handle of B.
2. **principal-to-secondary handle transformations**: each sub-assembly C (possibly a link) stores transformations $\mathbf{T_i^C}$ from its principal handle to its secondary handles H_i^C $(1 \leqslant i \leqslant k)$.
3. **child-to-parent transformations**: each internal sub-assembly A with parent C stores a transformation $\mathbf{T_A^C}$ from its principal handle to the principal handle of its parent.
4. **world transformations**: each sub-assembly C stores a transformation $\mathbf{T_C}$ from its principal handle to the global reference frame.

The various transformations are illustrated in Figure 1.

2.2 Recursive transformations updates

In our kinematic representation, the principal joint transformation of any joint is updated in constant time, based on the new joint configuration. It can be shown that the next two types of transformations can be recursively updated, from the leaves of the assembly tree to its root, when the joint configurations evolve.

Assume a linkage C is formed by assembling a linkage A with $l+1$ handles H^A, H_1^A, $\ldots$, H_l^A, and a linkage B with $m+1$ handles H^B, H_1^B, $\ldots$, H_m^B. The linkage C has $l+m$ handles: the principal handle of C is either a secondary handle of A or a secondary handle of B, while its $l+m-1$ secondary handles are the remaining secondary handles in A and B.

Now assume, without loss of generality, that the principal handle H^C of C is the secondary handle H_u^A of A, and let H_i^C denote a secondary handle of C. If H_i^C is also a secondary handle of A, say H_j^A, then $\mathbf{T_i^C} = \mathbf{T_j^A}(\mathbf{T_u^A})^{-1}$. If, however, H_i^C is a secondary handle of B, say H_j^B, then $\mathbf{T_i^C} = \mathbf{T_j^B}\mathbf{T_A^B}(\mathbf{T_u^A})^{-1}$. Moreover, the principal handle transformations can also be computed easily: the principal handle transformation $\mathbf{T_A^C}$ is actually equal to $\mathbf{T_u^A}$, and $\mathbf{T_B^C} = \mathbf{T_u^A}(\mathbf{T_A^B})^{-1}$. Finally, assuming the world transformation $\mathbf{T^C}$ of C is up-to-date, then $\mathbf{T^A} = \mathbf{T^C}\mathbf{T_A^C}$, and $\mathbf{T^B} = \mathbf{T^C}\mathbf{T_A^C}(\mathbf{T_A^B})^{-1}$. The case where the principal handle of C is a secondary handle of B is treated similarly.

2.3 Bounding transformations

The recursive computations presented above allow us to determine the positions and orientations of moving bodies over time. Once the principal joint transformations have been updated for a given time t (by, *e.g.*, evaluating sine and cosine functions for rotational joints), all the other transformations at time t are computed by multiplying 4×4 homogeneous matrices.

Our CCD uses these transformations and conservative *bounds* of them over progressively refined time intervals (*cf* Section 3). In order to efficiently compute these bounds, we use interval arithmetic [16, 22] to first bound the elementary functions in the principal joint transformations, and then perform interval counterparts of the matrix multiplications needed to compute the other types of transformations.

3 Continuous Collision Detection

Adaptive articulated-body dynamics [23] works by determining and simulating only a relevant subset of joints in the articulated body, which form a *sub-tree* of the assembly tree (the nodes above the dotted line in the assembly trees in Figure 2). Thus, at a given time step, the positions of only these nodes can change (*e.g.* for revolute joints, the corresponding angles). We now show how the kinematics representation introduced in Section 2 takes advantage of this fact to speed up the computation of the positions and bounds associated to the rigid bodies, and allows us to design a CCD algorithm which benefits from the adaptivity of the simulation. This algorithm shares some similarities with our previous work [22], but the key difference is in the

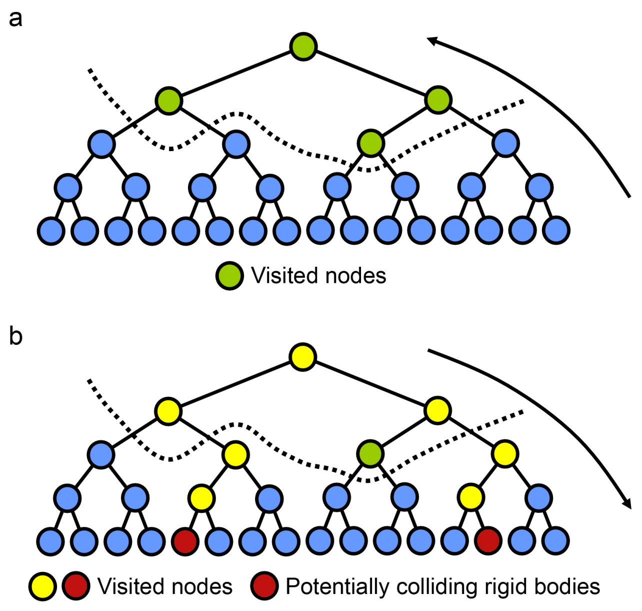

Figure 2: **Adaptive computation of rigid body transformations using an assembly tree.** *Only the nodes above the dotted curve are being simulated at the current time step, which allows us to limit the transformations updates to a limited number of nodes.* **a:** *updating the principal joint, the principal-to-secondary and child-to-parent transformations of the simulated nodes (green nodes).* **b:** *updating the world transformations for the potentially colliding rigid bodies (yellow nodes).*

computation of the positions and bounds on the rigid bodies and the exploitation of the adaptivity. In addition, we demonstrate self-CCD and CCD between multiple articulated bodies in Section 4.

Our continuous collision detection algorithm is composed of two main steps: a *body-level* culling step with axis-aligned bounding boxes (AABBs), and an exact contact computation step with hierarchies of oriented bounding boxes (OBBs; one hierarchy per rigid body).

3.1 AABB culling

We begin by computing bounds on the positions of all moving bodies over the current time interval, by recursively bounding the first three types of transformations of all active joints, from the bottom up (the green nodes in Figure 2.a). Once these bounds are updated, we bound the world transformations of *all* rigid bodies, by accumulating world transformations over *all* nodes. We then use these bounds to compute one AABB per rigid body, by multiplying the interval world transformations to the vertices of the root OBB which bounds the rigid body. This produces eight AABBs, each one bounding the trajectory of the OBB vertex over the whole time interval, and we compute the AABB that bounds these eight AABBs. By a simple convexity argu-

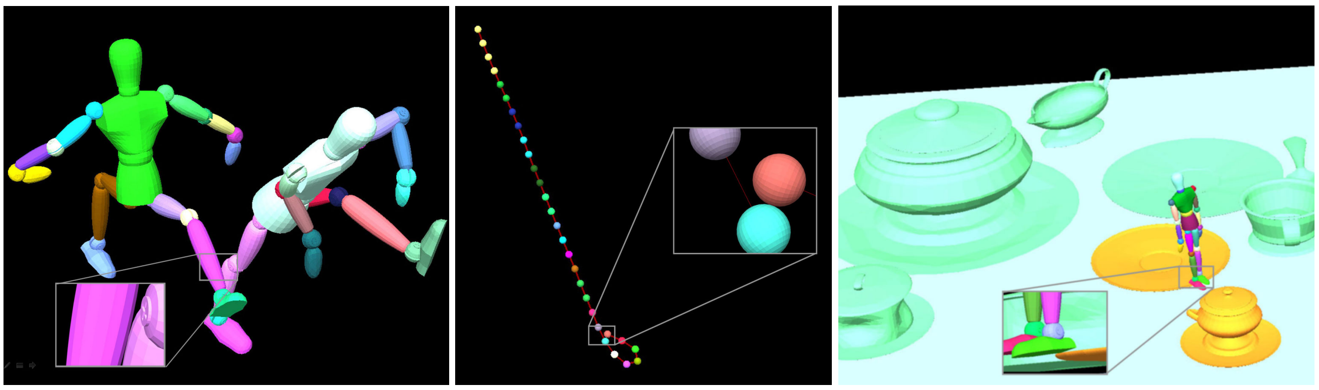

Figure 3: **Benchmarking examples. Left:** *a wooden man consisting of 29 rigid bodies, 16K triangles for each model, collides with another man. The number of active joints is 15.* **Center:** *a pendulum consisting of 30 rigid bodies and 16K triangles is self-colliding.* **Right:** *a wooden man falls due to the gravity and collides with static plates. The entire static environment consists of 44K triangles in total. In these figures, rigid bodies with identical colors belong to the same group of rigidified links.*

ment, this AABB bounds the rigid body over the current time interval. Note that this step is linear in the number of rigid bodies, but with a small constant as the bounds on the world transformations are computed only once per rigid body. The AABBs are then used to determine the pairs of potentially colliding rigid bodies (possibly within the same articulated body, or with rigid bodies in the static environment).

3.2 Computing the contact information

Once the potentially colliding rigid bodies have been determined, we compute the time of first contact and the contacting features using interval arithmetic. The key computation in this step is to evaluate the positions and orientations of the rigid bodies that might collide, as well as conservative bounds on these positions and orientations, over smaller and smaller time intervals. These are used to bound the trajectories of the OBBs (for efficient culling) and of the geometric features (vertices, edges and triangles, for precise contact time computation), of the potentially colliding rigid bodies. This is similar to step 3 in [22], but we can now perform those computations *adaptively*, based on the set of simulated joints.

Assume we want to determine the positions and orientations of the potentially colliding rigid bodies at a given time, or over a given time interval. As in the AABB culling step, we first update the first three types of transformations for all active joints, from the bottom up (*i.e.* the green nodes in Figure 2.a). However, we now compute the world transformations of the potentially colliding rigid bodies only (*i.e.* the red nodes in Figure 2.b). Thus, we accumulate the world transformations, top down, only on the way to these rigid bodies (*i.e.* the yellow and red nodes in Figure 2.b).

Assuming the assembly tree of an articulated body with n joints is balanced, an upper bound[2] on the complexity of computing the world transformations of k rigid bodies when m joints are active is thus $O(m + k\log(n))$. We show in the next section how this reduced complexity allows us to obtain potentially significant performance improvements.

4 Results

We have implemented our CCD algorithm under the framework of adaptive dynamics [23]. We highlight the performance of our algorithm in different benchmarking scenarios[3] such as *wooden men*, *swinging pendulum* and *a falling wooden man* by varying the number of active joints in the simulation (Figure 3). The benchmarking has been performed on a 2.19GHz AMD Opteron PC with 2GB RAM under Windows XP.

For the wooden men benchmark, a pair of wooden men characters are pulled together because of the spring attached between them. Initially, the wooden men models are placed at random configurations. For the pendulum benchmark, a pendulum consisting of many small balls swings because of gravity. In this benchmark, self-collision between a pair of balls is checked. For the falling wooden man benchmark, a wooden man is falling from the sky due to the gravity and colliding with obstacles such as pots and plates on the ground. For these benchmarking examples, their dynamics are governed by adaptive dynamics with different number of active joints, and the resulting CCD timings during collision are measured and averaged over several runs (*e.g.* 50). Figure 4 shows the resulting timings for each scenario.

[2] In practice, the complexity is smaller since the world transformations of the internal nodes are shared by multiple rigid bodies.

[3] The accompanying videos can be seen from *http://graphics.ewha.ac.kr/CCD4AD*

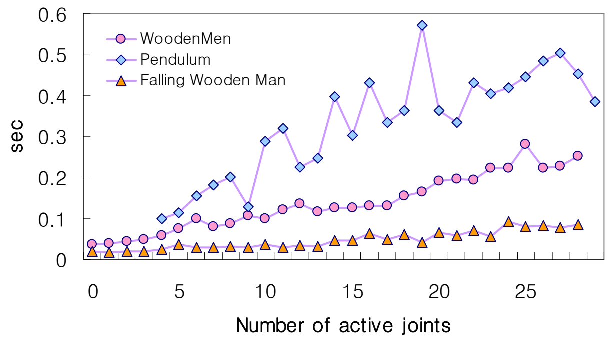

Figure 4: **Performance graph.** *The CCD time complexity gradually increases as a function of the number of active joints.*

5 Conclusion

This paper has introduced a CCD algorithm for an adaptive dynamics simulation of articulated bodies. This algorithm relies on a novel hierarchical representation of the kinematics of an articulated body, which can be selectively updated during the adaptive simulation. This allows us to reduce the complexity of the computation of the first time of contact, as well as of the contact information, when the dynamics of the articulated body are simplified. Our results demonstrate that, depending on the amount of dynamics simplification, this strategy leads to a potentially significant performance improvement. For future work, we plan to investigate several applications and extensions of this work, in particular to haptics and motion planning.

REFERENCES

[1] K. Abdel-Malek, D. Blackmore, and K. Joy. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*, 2002.

[2] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tiling for kinetic collision detection. In *Workshop on Algorithmic Foundations of Robotics*, pages 83–96, 2001.

[3] D. Bae and E. Haug. A recursive formulation for constrained mechanical systems dynamics: Part i. open-loop systems. *Mechanical Structures and Machines, Vol. 15, No. 3, pp. 359-382*, 1987.

[4] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 23–34. ACM SIGGRAPH, 1994. ISBN 0-89791-667-0.

[5] H. Brandl, R. Johanni, and M. Otter. A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. *IFAC/IFIP/IMACS Symposium, pp. 95-100*, 1986.

[6] J. F. Canny. Collision detection for moving polyhedra. *IEEE Trans. PAMI*, 8:200–209, 1986.

[7] Y.-K. Choi, W. Wang, Y. Liu, and M.-S. Kim. Continuous collision detection for elliptic disks. *IEEE Transactions on Robotics*, 2006.

[8] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer, Boston, MA, 1987.

[9] Roy Featherstone. A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 1: Basic algorithm. *International Journal of Robotics Research 18(9):867-875*, 1999.

[10] Roy Featherstone. A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 2: Trees, loops, and accuracy. *International Journal of Robotics Research 18(9):876-892*, 1999.

[11] J.M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-10, No. 11*, 1980.

[12] B. Kim and J. Rossignac. Collision prediction for polyhedra under screw motions. In *ACM Conference on Solid Modeling and Applications*, June 2003.

[13] D. Kim, L. Guibas, and S. Shin. Fast collision detection among multiple moving spheres. *IEEE Trans. Vis. Comput. Graph.*, 4(3):230–242, 1998.

[14] D. Kirkpatrick, J. Snoeyink, and Bettina Speckmann. Kinetic collision detection for simple polygons. In *Proc. of ACM Symposium on Computational Geometry*, pages 322–330, 2000.

[15] S. McMillan and D. E. Orin. Efficient computation of articulated-body inertias using successive axial screws. *IEEE Trans. on Robotics and Automation, vol. 11, pp. 606-611*, 1995.

[16] Ramon E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, New Jersey, 1966.

[17] Michael Ortega, Stephane Redon, and Sabine Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies. In *IEEE International Conference on Virtual Reality*, 2006.

[18] S. Redon, A. Kheddar, and S. Coquillart. An algebraic solution to the problem of collision detection for rigid polyhedral objects. *Proc. of IEEE Conference on Robotics and Automation*, 2000.

[19] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum)*, 2002.

[20] S. Redon, K. Kheddar, and S. Coquillart. Gauss' least constraints principle and rigid body simulation. *Proceedings of International Conference on Robotics and Automation*, 2002.

[21] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha. Interactive and continuous collision detection for avatars in virtual environments. In *Proceedings of IEEE Virtual Reality*, 2004.

[22] S. Redon, Young J. Kim, Ming C. Lin, and Dinesh Manocha. Fast continuous collision detection for articulated models. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, 2004.

[23] Stephane Redon, Nico Galoppo, and Ming C. Lin. Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3), 2005.

[24] Stephane Redon and Ming C. Lin. An efficient, error-bounded approximation algorithm for simulating quasi-statics of complex linkages. In *Proceedings of ACM Symposium on Solid and Physical Modeling*, 2005.

[25] Stephane Redon and Ming C. Lin. Practical local planning in the contact space. *In Proceedings of IEEE International Conference on Robotics and Automation*, 2005.

[26] F. Schwarzer, M. Saha, and J.-C. Latombe. Exact collision checking of robot paths. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, Dec. 2002.

[27] G. van den Bergen. Ray casting against general convex objects with application to continuous collision detection. *Journal of Graphics Tools*, 2004.

[28] Xinyu Zhang, Minkyoung Lee, and Young J. Kim. Interactive continuous collision detection for non-convex polyhedra. *The Visual Computer (Proceedings of Pacific Graphics)*, 22:9–11, 2006.