A New Library for Real-time Continuous Collision Detection[1]

Holger Täubig, Udo Frese
Cyber-physical Systems, DFKI Bremen, Germany

Abstract

We present the Kinematic Continuous Collision Detection Library (KCCD), a new library for real-time continuous collision detection for humanoid and industrial robots. The C++ library operates on joint state intervals. Such, it does not only test a single or N intermediate configurations but assures safety of a whole movement. The library uses sphere swept convex hulls (SSCH) as its volume representation, computes swept volumes and their distances for all body pairs of a robot, and provides an operation set that allows for a trade-off between accuracy and computation time. The paper gives an overview of the applied algorithm and describes basic features of the library, additionally provided configuration and visualization tools, as well as existing applications.

1 Introduction

We recently developed a new library for continuous collision detection based on fast swept volume computation as part of our project B-Catch [1, 2, 3], which we are going to make publicly available. The project B-Catch is involved with controlling a humanoid robot at high speed. For that reason we aimed for a self-contained, reactive collision detection module that handles significant braking distances geometrically precisely with limited computation time. Even though developed for a special purpose (safeguarding a humanoid robot only consisting of revolute joints) the library offers a variety of computation results, features such as conservative handling of measurement errors and motion, and varied robot configurations, i.e., different joint types. It is also very well suited to be applied in industrial environments as shown in **Fig. 1** for which all states of moveable parts or joints can be measured; perception itself is beyond the scope of the library.

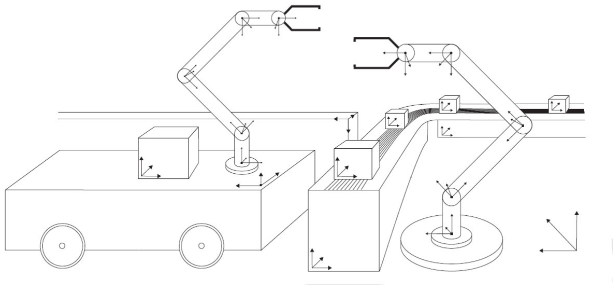

Figure 1: Example of an industrial environment in which the library can be applied.

In this paper we present the library for continuous 3d collision detection for industrial and humanoid robots and additional tools. The library will be available in C++. It is based on a special volume representation called *sphere swept convex hull (SSCH)*. SSCHs standouts include modelling of round and edged objects using very view points and less numerical problems due to the non-use of connectivity information such as triangle meshes.

Configuration files declare a kinematic tree consisting of the robot's defining joint frames and one or multiple bodies in each joint frame. After loading a configuration the state vector has to be updated cyclically and the collision detection function called. Each call reuses previous results and stops computation as early as possible taking into account which information is necessary. E.g., when looking for the smallest distance of all body pairs it will not compute a distance of a pair exactly if it is already known to be larger than another pair's distance. A more fancy reduction of computational effort is the distance update scheme which we apply.

The key feature of the library is to provide *continuous collision detection* by using an interval per joint as input (e.g., joint angle interval). The interval covers the whole braking motion instead of a single configuration for that joint. Potential collisions are computed for any configuration vector within the input interval vector. In that, a vector lies within a interval vector if each of its elements lies within the corresponding interval of the interval vector. This specification of the results that will be computed yields a conservative interpretation of the movement to be checked: it is the cross product of the intervals each joint moves within. This is appropriate for short and mid term movements, which is sufficient for reactive collision avoidance.

The library can provide different computation results: collision or not, the body pair having the smallest distance below a given safety margin, or the distances of all body pairs. In every one of these cases the distance of a body

[1]Research supported by the DFG under Grant No. FR2620/1-1 (BCATCH).

pair is the smallest possible distance of the two bodies during the input motion. Further, the library supports computing the collision model from a given Open Inventor model, i.e., computing an appropriate SSCH for each rigid body.
The rest of this paper gives an overview of the volume representation SSCH and the algorithm (Sec. 3), summarizes basic properties and features (Sec. 4), and shows existing application examples (Sec. 5).

2 Related Work

For algorithmic details and theoretical foundations as well as a thorough discussion of related work we refer to [1]. We would like to highlight [4] and [5], these textbooks provide a very good and comprehensive overview of real-time collision detection methods.
There are other collision detection libraries available such as PQP[6], SWIFT[7], V-Clip[8], and SOLID[9], but their focus differs from the particular robotics focus of our new library. A major functionality of the KCCD library is the processing of the robot kinematic which provides the real-time swept volume computation. The computation of the distances between these swept volumes is another part, in the KCCD library essentially based on a GJK[10] implementation. This second part compares to the functionality of the before mentioned libraries. However, we did not apply them because they usually operate on triangle mesh representations and use bounding volume techniques to achieve a very good performance[6, 7]. As a consequence, on the one hand they are able to handle complex objects consisting of thousands of triangles but on the other hand operate on rigid objects and usually require a preprocessing for the construction of triangle meshes and bounding volume hierarchies that is rather computationally expensive. But the swept volumes to be checked for collision vary in successive time steps, i.e., the volume shape changes more than only its pose, which would make the preprocessing computation necessary in each time step. Furthermore, these libraries cannot be applied for the task as a whole, thus including swept volume computation, as to the knowledge of the authors the libraries do not support continuous collision detection but operate on single states for their objects in each time step.

3 Overview of SSCH and the Collision Detection Algorithm

The library uses sphere swept convex hulls (SSCH) as its volume representation. A SSCH is given by a finite set of points $P = \{[p_l]_{l=1}^{n}\}$ and a radius r. Its static volume is defined as

$$\mathcal{V}(r;P) = \operatorname{conv} P + \{b \in \mathbb{R}^3 \mid |b| \leq r\}, \tag{1}$$

where $\operatorname{conv} P$ is the convex hull of a point-set P.
So each volume is the Minkowski-sum of a convex polyhedron given by a set of points, and a ball of radius r. Hence, it can bound both edged and round bodies tightly with few points. Even though the name may suggest something different, a SSCH is at first a representation of a static volume, e.g. a body of the robot. It is internally stored as point list P and radius r and does not contain any robot motion so far; this is realized by the algorithm in terms of computing appropriate P and r. P and r do not contain any connectivity information and the convex hull $\operatorname{conv} P$ does not have to be computed explicitly, which avoids typical problems of computational geometry involving degenerate triangles.
The robot's configuration consists of a kinematic model defining joint-frames and a geometrical model with the robot's rigid bodies each represented as a SSCH in one joint-frame. Non-convex bodies may be represented by multiple volumes as an SSCH is always convex. The kinematic model currently can consist of revolute joints and prismatic joints.
At its heart, the library is essentially involved with swept volume and distance computation of all relevant body pairs for an input state interval vector. However, we show the full collision avoidance algorithm from [1] in the following as this covers an important use case of the library. It computes the input interval vector covering the braking behaviour and measurement errors (step 1), then calls the basic library function (which executes step 2 and 3), and finally responds to the results appropriately (step 4). The swept volume of one of the robot's bodies is the volume covered by this body during the considered motion. The algorithm operates as follows:

1. **Compute all joint intervals** $\mathcal{Q} = [q^0; q^1]$, such that when the robot starts braking the next cycle, it will stop within this interval. The intervals are based on joint angles, joint angle velocities, latency, and worst-case deceleration, as well as joint angle uncertainties.

2. **Compute swept volumes $\mathcal{V}_k^i$ of all bodies B_i in all joints J_k from the body down to the robot base by successively including the sweeping effect of one joint J_k after the other.** The swept volumes are represented in coordinates of the corresponding joint-frame C_k of joint J_k.

3. **For each body pair** (B_i, B_j)

   Compute the distance of $\mathcal{V}_k^i$ and $\mathcal{V}_k^j$ in the first common joint-frame C_k on the sequences of joints from B_i and B_j down to the robot base.

4. **Stop robot if any of the distances from 3) is zero.**[2]

Further details and theoretical foundations can be found in [1].

[2] Alternatively, less or equal a configured *safety distance*.

4 Features

We will now summarize basic properties and features of the library.

4.1 Conservative & Real-Time Computation

The library provides a strictly conservative and geometrically precise collision detection or rather distance computation algorithm. The distance of a body pair therein is the smallest possible distance of the two bodies during the considered motion, or precisely for any configuration vector within the input interval vector. In fact, it computes a lower bound of the distances of all body pairs in a configurable amount of computation time, thus in real-time. Similar to tother any-time algorithms, if the bound computed within the time budget is not good enough to rule out a collision, the robot has to stop, maybe unnecessarily. The time budget that is necessary in practice depends on the information to be computed and the robot configuration. Depending on the information needed the algorithm saves as much computation time as possible. Different computation results in ascending order of necessary time budget are: collision or not, body pair having the smallest distance below a given safety margin, or distances of all body pairs. The influence of the robot configuration has two aspects: First, the complexity of the robot's kinematic tree has an inherent influence on the computation time. Second, the library offers a set of operations (multiple versions of each joint type) that allows for a trade-off between accuracy and computation time within robot configuration.

4.2 Operate in the LCA-Frame

As a major task the library computes, respectively bounds, the distance of the swept volumes of two bodies. Both of these swept volumes as well as their distance depend on the frame they are computed in. In particular, the effects of the frame onto both swept volumes do not necessarily cancel each other out. We showed this general property of swept volumes in [1] and inferred that "for two bodies [...] we can choose any frame on the connecting path [in the kinematic tree] to compute and intersect swept volumes". As a consequence, the simplest and probably most efficient solution is to operate in the lca-frame of a body pair. The lca-frame is the least-common-ancestor in the kinematic tree, thus the deepest node, i.e., joint frame, in the tree that is a parent of both bodies. This is done in the library: for every body pair it computes both swept volumes and their distance in the lca-frame of the body pair.

4.3 Supported Joint Types

The kinematic tree is build from separate joints during configuration of the robot. The library currently supports prismatic joints and revolute joints completely. Complete support means providing multiple conservative implementations per joint type. These are: a fast calculation of the joint without state intervals (interval width is zero), and two versions of the joint with intervals. One of the interval versions is more accurate and computationally expensive, the other one is faster and generates a larger approximation error. But both versions are strictly conservative. The robot configuration defines which one is chosen. In this way, the user can balance between accuracy and computation time. Further details on prismatic and revolute joints can be found in [1].
Current work in progress is the integration of so called vehicle joints into the library. Vehicle joints allow to model vehicles and their braking behaviour as shown in [11], where equivalent techniques and representations have been used for 2D collision detection for vehicles. The result will be an integrated 2D/3D collision detection for vehicles with manipulators or industrial settings as in Fig. 1.
For practical purposes we recently also added joints that cover arbitrary 3D translation and rotation. They differ from prismatic respectively revolute joints in that they use 3 dimensions in input state vector instead of having a configured fixed axis and just on degree of freedom. They can be used to position and orient moveable objects arbitrarily in the scene (position and orientation must be measured and given as part of the state vector) but they do not provide a motion representation consisting of 3 dimensional intervals. Instead, uncertainty in terms of a single valued error bound can be configured. This is meant to be the upper bound of the measurement error, which is the maximum distance to the true position for 3D translations and the maximum rotation angle between true and measured orientation for 3D orientations. Again, implementations are conservative. Resulting volumes cover all states within the configured uncertainty bounds.

4.4 Configuring the Collision Volumes

One of the time consuming and safety-critical issues within configuration is the creation of a SSCH representation for each rigid body of the robot. As most users own a CAD model of their robot, we alleviate that configuration step by providing a automatic volume optimization tool that computes a conservative SSCH for a given Open Inventor model. The Open Inventor model has to contain exactly one rigid body and the user can choose the number of points used by the SSCH. Complex CAD models have to be split into separate files containing just one rigid body per file, other CAD models converted to Open Inventor before the tool can be applied.

4.5 Visual Configuration Inspection

Our library is thought to provide a part of the software of a safety device. Even though it is not certified according to an appropriate standard so far, we nevertheless prepared it to cover the features that would be necessary for such a

certification. One of these features required by norms such as IEC 61508 is to provide a instrument for the user to check the correctness of the configuration. State-of-the-art in comparable domains such as safety laser rangefinders is the visual inspection of the configured safety zones; in our case that means visualization of the collision model. An additional tool provides that by overlaying the collision volumes into the live image of a camera that has an arbitrary but fixed position. This is done while the robot is moving, so the tool has to be provided with the current state vector.
In the beginning of that visualization the camera pose is determined semi-automatically by a simple labelling process. The accuracy of this labelling was $0.3°$ orientation error and $1cm$ distance error for the camera pose within our original application (cf. Sec. 5.1).

5 Applications

5.1 Humanoid Robot Justin

Fig. 2 presents results from the original application of the library. It is used to safeguard DLR's humanoid Justin in the B-Catch project, whose task is to catch two simultaneously thrown balls [2, 3]. On the left Justin and its collision model (green volumes) are shown. The 26 bodies of Justin's collision model (20 joints) only contain 80 points and 26 radii. A single body's representation uses only 3.1 points on average, less than 4 for a tetrahedron, the simplest polyhedral volume. Nevertheless, the collision model is tight and conservative. The bounded computation time needed by the library to apply this model is 0.4ms on a INTEL T2500@2GHz processor.
Fig. 2 right provides the swept volumes that arised in an experiment of moving Justin's hands towards each other with different velocities until the safety module brakes. It shows the colliding swept volumes at the time braking is triggered. The swept volumes grow with the velocity because the braking distance (the considered movement) grows with the velocity.

5.2 Industrial Application

Within the project INVERITAS the library is used to prevent collisions of their demonstrator. INVERITAS is a joint project including the partners EADS-ASTRIUM, Jena-Optronik and the Robotics Innovation Center of the DFKI. The project is involved with the prototypic realization of a broad-spectrum rendezvous and capture system. The library is used to protect their physical technology demonstrators consisting of an six-axis industrial robot and a cable-guided 3D-movement system in the environment of DFKI's exploration hall. **Fig. 3** shows this example of an industrial setup. The six-axis industrial robot and its payload, a satellite, is configured using prismatic and revolute joints. The location of the cable guided 3D-movement system is configured using 3D translation and rotation joints. Its pose is received from the exploration hall system. The exploration hall environment will also be modelled but is not shown in the figure. The shown configuration uses 50 points and 13 radii for 13 bodies in a 12 DOF system. Still, this is a rather small example but the library is capable of larger ones.

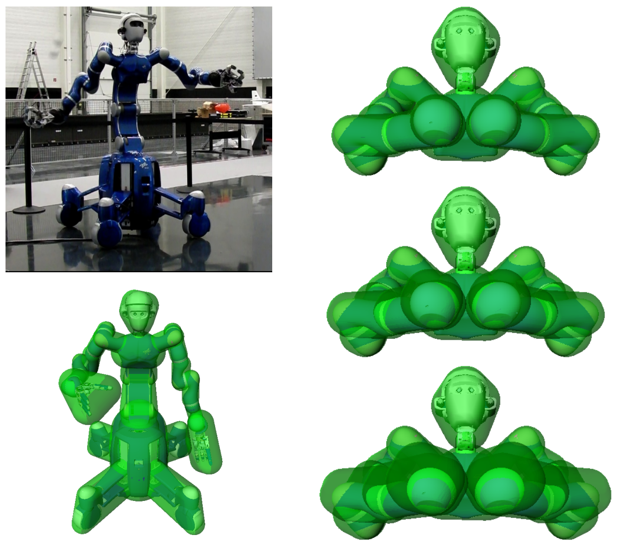
Figure 2: *(left)* DLR's mobile humanoid Justin and its collision model. *(right)* Experiment of moving Justin's hands towards each other with 0.22, 0.65, and $1.1\mathrm{m/s}$ *(top to bottom)* until the safety module brakes [1]. Figures show the colliding swept volumes at the time braking is triggered. The hands are omitted here for better visibility.

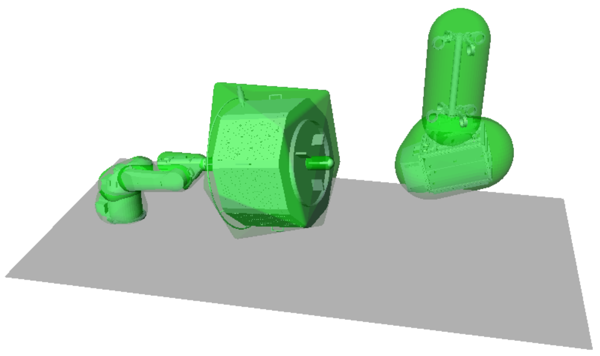
Figure 3: Collision detection setup in the INVERITAS project consisting of an six-axis industrial robot, its payload (a satellite) and a cable-guided 3D-movement system (cable not included).

5.3 Distance of All Body Pairs Example

The library has further been used for reactive self collision avoidance [12], where it provides the N body pairs having smallest distance to an appropriate control law. For each of these body pairs their distance and potential collision points, i.e., the points on the two bodies that have minimum distance, were computed. This application is an example for a different computation result provided by the library. It also used DLR's humanoid Justin and its collision model shown in Fig. 2.

6 Summary

We presented a new library for real-time continuous collision detection based on fast swept volume and distance computation. We described basic features, additional configuration and visualization tools, and currently existing applications. The library will soon be available online at `www.dfki.de/cps/3d-collision-avoidance`.

References

[1] H. Täubig, B. Bäuml, and U. Frese, "Real-time swept volume and distance computation for self collision detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011, pp. 1585–1592.

[2] B. Bäuml, O. Birbach, T. Wimböck, U. Frese, A. Dietrich, and G. Hirzinger, "Catching flying balls with a mobile humanoid: System overview and design considerations," in *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Bled, Slovenia, 2011, pp. 513–520.

[3] B. Bäuml, F. Schmidt, T. Wimböck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. Grebenstein, O. Eiberger, and G. Hirzinger, "Catching flying balls and preparing coffee: Humanoid Rollin'Justin performs dynamic and sensitive tasks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011, video.

[4] C. Ericson, *Real-Time Collision Detection*, ser. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, 2005.

[5] G. van den Bergen, *Collision Detection in Interactive 3D Environments*, ser. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, 2003.

[6] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, USA, 2000, pp. 3719–3726.

[7] S. A. Ehmann and M. C. Lin, "Accelerated proximity queries between convex polyhedra by multi-level voronoi marching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan, 2000, pp. 2101–2106.

[8] B. Mirtich, "V-clip: fast and robust polyhedral collision detection," *ACM Transactions on Graphics (TOG)*, vol. 17, no. 3, pp. 177–208, July 1998.

[9] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *Journal of Graphics Tools*, vol. 2, no. 4, pp. 1–14, April 1997.

[10] 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*, vol. 4, no. 2, pp. 193–203, April 1988.

[11] H. Täubig, U. Frese, C. Hertzberg, C. Lüth, S. Mohr, E. Vorobev, and D. Walter, "Guaranteeing functional safety: design for provability and computer-aided verification," *Autonomous Robots*, vol. 32, no. 3, pp. 303–331, April 2012.

[12] A. Dietrich, T. Wimböck, H. Täubig, A. Albu-Schäffer, and G. Hirzinger, "Extensions to reactive self-collision avoidance for torque and position controlled humanoids," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 3455–3462.