The Journal of Open Source Software

robot_collision_checking: A Lightweight ROS 2 Interface to FCL (Flexible Collision Library)

Mark Zolotas [1*¶], **Philip Long** [2*], and **Taskin Padir** [1,3]

1 Northeastern University, USA (at the time of this work) **2** Atlantic Technological University, Ireland **3** Amazon Robotics, USA ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.07473

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @Aravind-Sundararajan
- @cadojo

Submitted: 15 August 2024
Published: 27 January 2025

Summary

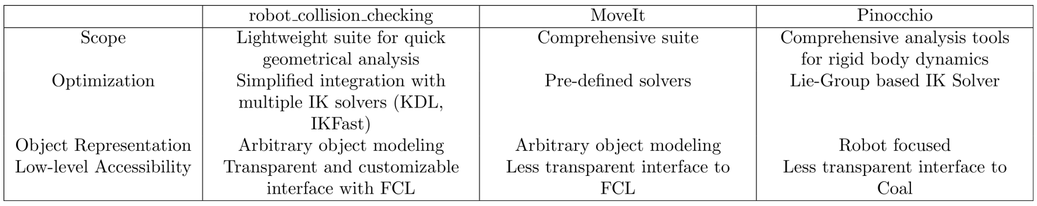
This paper presents robot_collision_checking, a C++ library that provides a Robot Operating System (ROS) (Quigley et al., 2009) interface to the Flexible Collision Library (FCL) (Pan et al., 2012) for typical robotics applications. FCL is an open-source C++ library that provides efficient collision detection and distance computation for 3D environments. While these capabilities are crucial in robotics to ensure safety and enable effective motion planning, FCL is not readily available for many robot architectures built atop ROS. Given that the robotics community widely relies on ROS as the standard for software development, it would greatly benefit from a lightweight ROS interface to FCL. The robot_collision_checking package fulfils this demand by exposing FCL functionality to ROS message types, thereby allowing robotics researchers and practitioners that rely on ROS to easily access the collision and distance checking features of FCL.

The robot_collision_checking package can calculate collisions and distances between a variety of collision objects, including solid primitives (spheres, boxes, cylinders), planes, meshes, voxel grids, and octrees (via the OctoMap library (Hornung et al., 2013)). Collision worlds that contain multiple collision objects can also be created and maintained. This enables collision and distance checking between single objects, as well as entire collision worlds. The robot_collision_checking package includes an example node that demonstrates how to create a collision world of ROS objects, use FCL functionality to perform collision-checking on these objects, and visualize the world in RViz (Kam et al., 2015), e.g., for debugging purposes.

Additionally, we include ROS 1 and ROS 2 (Macenski et al., 2022) implementations of the robot_collision_checking package. There are notable differences between these implementations due to the differences between ROS 1 and ROS 2, as well as how collision objects are handled by the collision world class and utility functions of the core C++ library. As the ROS 2 version is up-to-date, more well-documented, and continues to receive ongoing support, we encourage users of robot_collision_checking to opt for this implementation. End-users may also test the Docker image available to the code repository if they wish to explore this package without installing ROS on their machine.

Statement of Need

Collision-checking is an increasingly important tool as robots are deployed into unstructured and dynamic environments, while ROS 1 and ROS 2 provide the most popular means of controlling robots for research applications. In the ROS ecosystem, one popular means of enabling collision-checking is via MoveIt (Coleman et al., 2014), a path planning and trajectory execution open-source software. The MoveIt collision-checking API can expose two different collision checkers: bullet and FCL (Pan et al., 2012). However, to leverage this functionality users have to install the entire MoveIt suite and either integrate their robot into MoveIt or ensure that their platform is already available to the software suite. Moreover, while MoveIt is an extremely sophisticated motion planning library, accessing lower-level functionality for collision and distance checking requires in-depth knowledge of the library's structure and hierarchy. Pinocchio (Carpentier et al., 2021) is another powerful robot modeling software that is also built upon FCL (a specific variant, known as Coal) but suffers from the same overhead as MoveIt. The `robot_collision_checking` library aims to address the need for a lightweight alternative by providing a simple and transparent ROS interface to the FCL library. A comparison between `robot_collision_checking`, MoveIt, and Pinocchio is summarized in Figure 1.



	robot_collision_checking	MoveIt	Pinocchio
Scope	Lightweight suite for quick geometrical analysis	Comprehensive suite	Comprehensive analysis tools for rigid body dynamics
Optimization	Simplified integration with multiple IK solvers (KDL, IKFast)	Pre-defined solvers	Lie-Group based IK Solver
Object Representation	Arbitrary object modeling	Arbitrary object modeling	Robot focused
Low-level Accessibility	Transparent and customizable interface with FCL	Less transparent interface to FCL	Less transparent interface to Coal

Figure 1: Table comparing `robot_collision_checking`, MoveIt, and Pinocchio.

Our package is similar to Python-fcl, which provides a Python binding of FCL that could also be used in a ROS architecture. The main distinction is that our implementation is written in C++, providing enhanced computational efficiency thanks to its compiled nature. Furthermore, our package includes convenience functions to directly interact with ROS/ROS 2 messages and easily display results in RViz. The ros_collision_checking package also offers a collision-checking system for 2D vehicles in a ROS environment. Our collision-checking system instead extends the general capabilities of FCL for proximity querying any geometric model and can thus be applied in numerous robotics contexts where proximity information about the 3D environment is beneficial.

The interface supplied by `robot_collision_checking` is especially practical for obstacle avoidance and path planning in robotics use-cases. First, the robot's collision geometry (e.g., extracted from its URDF model) and surrounding objects perceived by the robot's sensors (e.g., an OctoMap representation given depth data) can be added to a collision world that is constructed and maintained through our `robot_collision_checking` interface. The resulting collision and distance information exposed by this interface then enables safe plans for the robot's motion to be generated using standard motion planning algorithms. For instance, this package could be employed to compute virtual repulsive forces based solely on the robot's tool pose, thereby enabling a potential fields method to navigate the tool through an environment without requiring MoveIt's installation overhead. Lastly, the `robot_collision_checking` package facilitates rapid checks for distances and collisions between arbitrary objects and is not solely limited to robotic components. This versatility is particularly valuable for applications involving complex environmental analysis, such as those encountered in human-robot interaction.

The `robot_collision_checking` library is currently being used by the constrained_manipulability package, a motion planning framework for robot manipulators developed by the same authors. Within the `constrained_manipulability` package there are more examples of using the `robot_collision_checking` library with URDF files and collision meshes to calculate collisions and/or distances between a robot and environmental objects, such as primitives and OctoMaps.

Future Work

Our objective is to maintain `robot_collision_checking` in a manner that ensures seamless integration with future advancements in the core FCL library (Pan et al., 2012). As FCL continues to evolve, introducing new collision-checking functionality, support for more complex geometric representations, or other enhancements, we aim to maintain our package and capitalize on these developments. This forward-looking approach ensures that `robot_collision_checking` remains robust, versatile, and aligned with state-of-the-art collision detection software.

Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be constructed as a potential conflict of interest.

Acknowledgements

Mark Zolotas is currently at Toyota Research Institute (TRI), Cambridge, MA, USA. This paper describes work performed at Northeastern University and is not associated with TRI.

Taskin Padir holds concurrent appointments as a Professor of Electrical and Computer Engineering at Northeastern University and as an Amazon Scholar. This paper describes work performed at Northeastern University and is not associated with Amazon.

References

Carpentier, J., Budhiraja, R., & Mansard, N. (2021, July). Proximal and Sparse Resolution of Constrained Dynamic Equations. *Robotics: Science and Systems 2021*. https://hal.inria.fr/hal-03271811

Coleman, D., Sucan, I., Chitta, S., & Correll, N. (2014). Reducing the barrier to entry of complex robotic software: A moveit! Case study. *Journal of Software Engineering for Robotics*, *5*(1), 3–16.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. https://doi.org/10.1007/s10514-012-9321-0

Kam, H. R., Lee, S.-H., Park, T., & Kim, C.-H. (2015). Rviz: A toolkit for real domain data visualization. *Telecommunication Systems*, *60*, 337–345.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. https://doi.org/10.1126/scirobotics.abm6074

Pan, J., Chitta, S., & Manocha, D. (2012). FCL: A general purpose library for collision and proximity queries. *IEEE International Conference on Robotics and Automation*, 3859–3866. https://doi.org/10.1109/ICRA.2012.6225337

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. (2009). ROS: An open-source robot operating system. *ICRA Workshop on Open Source Software*, *3*.