View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Robotic Proximity Queries Library for Online Motion Planning Applications

Xavier Giralt, Albert Hernansanz, Alberto Rodriguez and Josep Amat
Technical University of Catalonia
Spain

1. Introduction

One of the most important problems to solve in robotics is the collision avoidance between a robot and its environment. A robot should perceive the risk and have a reactive behaviour before an imminent collision occurs. Path planning is a hard computational problem, so having a fast tool to calculate collisions is a key factor to decrease the necessary time to generate safety trajectories. In applications where no path planning exists, for instance in manual guidance or teleoperation, a real time collision detector is needed so as to avoid collisions and to be able to interact with the environment, for example sliding over a surface. The knowledge of minimum distances between robots or objects that share a workspace enables robots to behave in a predictive way. In the human-robot interaction field, virtual fixtures can be used both to prevent collisions and help the human operator by increasing his performance. In this kind of applications minimum distance and collision detection must be known in real time.

A new library: Robotic Proximity Queries (RPQ) package has been developed to deal with these requirements, using PQP as the proximity query engine. The original package has been used to optimize the queries when working with open kinematic chains, like robots. These optimizations have been done with the aim of improving the time performance of the generic package and simplifying its use in robotic environments. A system composed of two robots has been used as a test bed to show the performance of the RPQ library.

Two applications that benefit from RPQ performance are presented. First, a robotic assisted surgical application, with an assisted cut of a rigid tissue. The surgeon guides freely the driller held by a slave robotic arm that avoids undesired drillings by means of virtual protections. The second application is a dynamic expansion of the working space by means of multirobot cooperation. With these applications, not only proximity queries are shown, but also the graphical interface and the use of a virtual robot based on RPQ.

2. Related Work

During the last years, great efforts have been devoted to the development of efficient collision detection algorithms due to their wide range of applications, such as CAD/CAM, manufacturing, robotics, simulation and computer animation. A wide study of the performance and applicability of such methods can be found in (Geiger, 2000). Proximity

query algorithms vary in terms of their range of applicability and the type of queries they can solve, mainly collision detection, minimum distance computation and interpenetrations modelling. Although most algorithms allow as input triangulated meshes of 3D points, they differ in the way those points are pre-processed and represented internally in order to speed up specific queries.
There is a wide set of methods that rely on Lin-Canny or Gilbert-Johnson-Keiethi like algorithms for computing minimum distances between pairs of objects as I-Collide, Swift, Swift++, SOLID, DEEP…, but they are only applicable to convex polytopes (Ehmann et al., 2000),(Ehmann et al., 2001) and (Kim et al., 2002). This restriction makes them inappropriate for RPQ purposes, due to the need to deal with more general geometric models.
More general collision detection methods usually base their efficiency on pre-computed representations by means of hierarchies of bounding volumes. Their differences rely on the specific type of bounding volumes used, ranging from binary space decompositions, spheres trees to oriented bounding boxes (OBB).
Among this set of algorithms, RAPID and PQP turn to be those that have both, fewer restrictions in the range of allowable geometric models and an easier application programming interface (API). Both of them use oriented bounding boxes for performing collision tests, and have similar time performances. However, the fact that PQP offers a wider range of queries, including minimum distance computation and tolerance tests makes PQP the best option for the proximity queries engine of RPQ, the Robotics Query Package presented in this paper.

3. Library Description

The goal of the Robotic Proximity Queries (RPQ) library is to offer an easy, modular and fast proximity query package oriented to robotics. As explained above, the aim of the project is not the development of a new collision detector, but specialize an existing one into the robotics field.
As described in the previous section, there is a wide set of general purpose proximity query packages. The criterions used to choose PQP as the best candidate for the development of RPQ are:

-Types of proximity queries available.

-High time performance on proximity queries.

-Ability to use geometrical models based on triangulated meshes of points.

-Lack off restrictions on possible geometrical models.

-Open source library.

-Easy API.

The PQP library has been developed by UNC Research Group on Modelling, Physically-Based Simulation and Applications and offers three different kind of queries:
-Collision detection: detecting whether two models overlap, and optionally, give the complete list of overlapping triangle pairs.
-Distance computation: computing the minimum distance between a pair of models.

-Tolerance verification: determining whether two models are closer or farther than a given tolerance distance.

RPQ has been implemented in C++ language and its graphical interface has been developed using OpenGL. The RPQ library can be easily integrated into any software application.
The library interface allows non expert programmers to use it in an easy manner. The graphical interface is a separate module, allowing the programmer to decide whether using it or not. Fig. 1. shows the integration of the library and its graphical interface into a generic application.

3.1 Class Implementation

The RPQ library is based on the Object Oriented paradigm. Focused on this paradigm, and based on robotic environments, three main classes have been developed: Scenario, Object and Robot.

Scenario
Scenario is the workspace where the objects cohabit. Concerning its implementation, Scenario is a class that contains all the objects (Robots and generic objects), a global reference frame, and all the methods necessary to generate the proximity query.

Object
An Object is the minimum entity that exists in a Scenario. There are two types of Objects: simple and complex. A simple Object is represented by a geometrical model composed of a set of triangles referred to a frame tied to the Object. The Object has also a transformation matrix to refer itself to the world reference frame. A complex Object is an Object composed of a set of geometrical models with joints (rotational or prismatic) between them. Thus, a complex Object is an open kinematic chain composed of sub objects. The transformation matrix M_i refers $subobject_i$ to $subobject_{i-1}$. The transformation matrix M_0 refers the object base $subobject_0$ to the world. The object stores its own geometrical model. Concerning its implementation, an Object is a class containing the geometrical model, the transformation matrix and a set of methods to position and to orient itself in space. This class also contains methods to calculate the different detail representations of its geometrical model.

Robot
A Robot is a particularization of a complex Object where each of its links is represented by a simple Object. A Robot has a set of functions to make a complex Object as similar as possible to a real robot. For instance, the spatial relationship between links is described using the Denavit-Hartenberg notation. Direct and inverse kinematics can be calculated considering the robots own restrictions (joint limitations, configurations, etc). Concerning implementation, the class Robot is derived from the class Object. Robot adds all the functions that are necessary to control a robot. For instance joint positioning of the robot (direct kinematics), position and orientation of its tool center point (inverse kinematics), change of the robot configuration, joints overshoot…These added functions with respect an Object are very helpful when a new robot is created or used in robotic applications like simulators, path planners, etc.

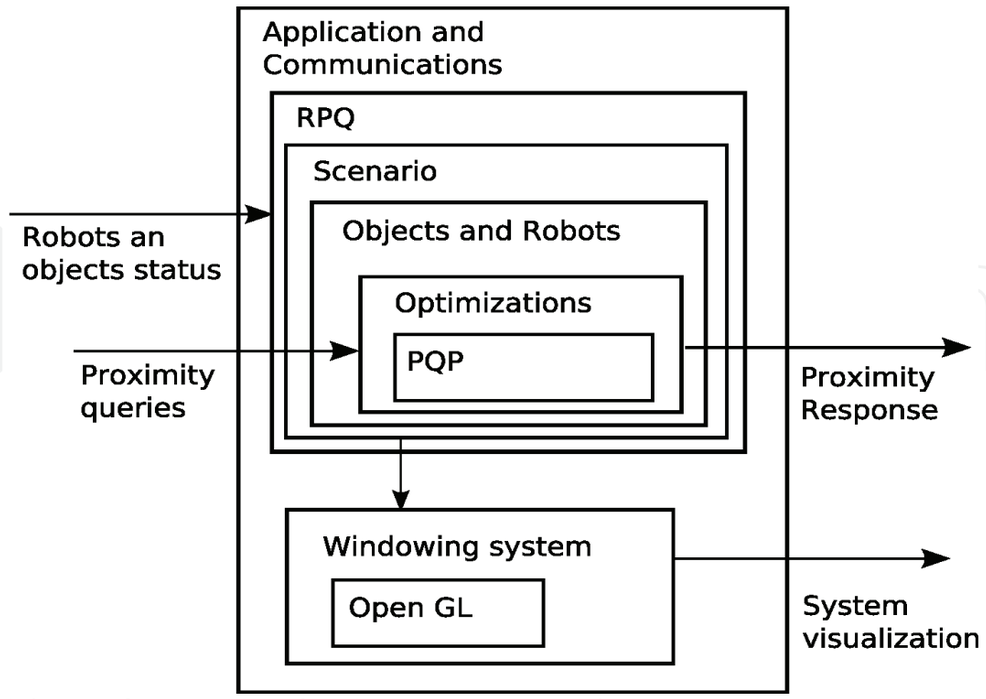

Fig. 1. Schema of integration of RPQ into a generic application

3.2 Optimizations

PQP is a generic package that does not use the knowledge of the object's kinematics. In contrast, RPQ is oriented to robotics, and the knowledge of robot's kinematics is the base of the optimizations that specialize it. RPQ is designed to answer proximity queries between two robots or between a robot and any kind of rigid object. Three optimizations have been developed and tested to improve the performance offered by PQP.

Different resolution levels of object's representation

Objects can be represented in very different resolution levels. The idea of this optimization is to use the simplest representation models (minimum number of triangles) to discard collisions. The lower the number of triangles of the geometric model is, the faster the collision queries are executed.

Three resolution levels are used to represent robots and two for the rest of objects. The highest resolution level is the complete geometrical model. The second level is the oriented bounding box (OBB) of each sub object in which a complex object is divided. The lowest resolution level is the bounding box of the whole complex object. This level is only defined for complex objects with more than a sub object, as in robots with several links. There are other possible intermediate resolution levels that can be used, for instance the convex hull. It offers a good ratio between resolution and the quantity of triangles, although it does not reduce it as drastically as the low resolution levels chosen.

This optimization is useful in two different situations. First, in applications where no high precision is required, for instance when the precision of the OBB or the convex hull of each link is enough. The second situation occurs when the different resolution levels are used in a complementary manner.

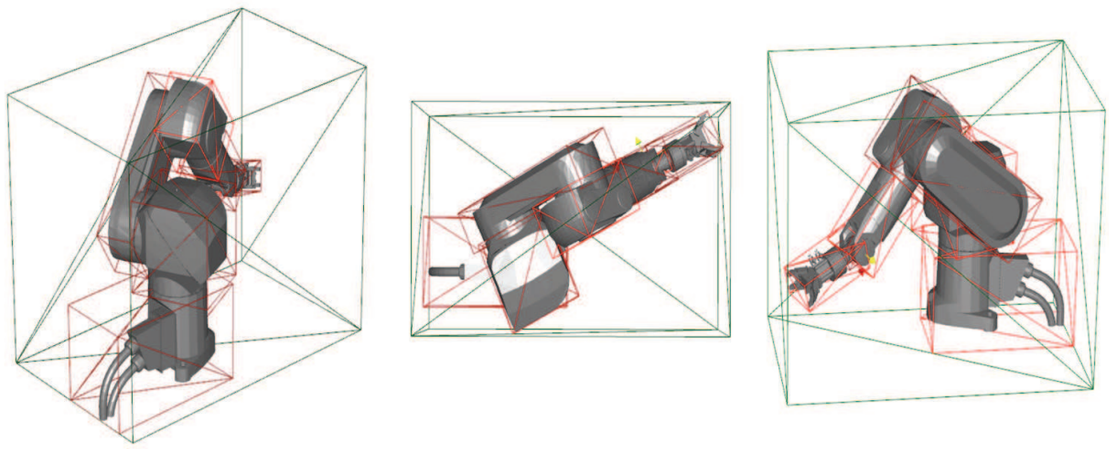

Fig. 2. Robot with three resolution level representation: The geometrical model of each link (L3), the OBB of each link (L2) and the whole bounding box(L1).

When a collision query is performed, a low to high resolution level list of collision queries is generated. Starting with the lowest resolution level, queries are generated until any collision can be completely discarded. For instance, if a possible collision between two 6 DOF robots is studied, the first query is done between the bounding boxes of each robot. If the collision can not be discarded, then the bounding box of each link is used. If at this level collisions still can not be discarded, the geometrical models of each link are checked. Fig. 2. shows a Robot with its three resolution levels of representation.
A test has been developed for the evaluation of the performance of the optimizations. It consists on a couple of virtual robotic arms (Staubli RX60B) that are placed one close to the other in a scenario. The geometrical models used for the test are high resolution (composed of 23012 triangles each). The robots are placed at a variable distance between them and the scenario is checked for collisions for an equidistributed set of joint positions in their 6 DOF. This test allows us to study the dependency on the performance of the proposed improvements in terms of the probability of collision.

Collision queries sorted using a Weight Matrix
This optimization is based on two assumptions. The first one is that the goal of the query is just to know whether there is a collision or not, but not the number of them. The second assumption is that the kinematics and the morphology of the robots are well known.
Given these assumptions, the objective is to find quickly whether there is collision or not, by means of minimizing the number of partial collision queries. The knowledge of the kinematics and the morphology of the robots give us the possibility of assigning an a priori collision probability to each link of the robot with respect to the rest of robots and objects present in the same workspace. During execution time, these probabilities are automatically updated depending on the result of the collision queries (Probability increases in case of detecting a collision and decreases otherwise). Therefore, a weight matrix C is generated combining the probability of collision between each pair of objects in the workspace. These weights determine the order of the collision queries. A simple way to assign a collision probability to the links of a robot is to assign higher probability to those links that are farther in the kinematic chain, with respect to the base of the robot.
Collision Matrix

The Collision Matrix is a binary matrix that reflects the possibility of collision between two objects. If the collision matrix indicates that a collision between two objects is impossible, its correspondent collision query is not performed. Of course, a matrix query is much less expensive than a collision query in computational terms. This optimization improves the performance of the system when a high number of collision queries are discarded by the Collision Matrix.
The performance of the Collision Matrix has been studied using the same test designed for testing the use of different levels of representation. The results are shown in Fig. 3. The farther the robots are, the lower is the number of links that can collide, and therefore, the higher is the number of queries that are solved with the Collision Matrix. As it is shown in the figure, using the Collision Matrix the number of collision queries decreases, so does the time to solve the query.

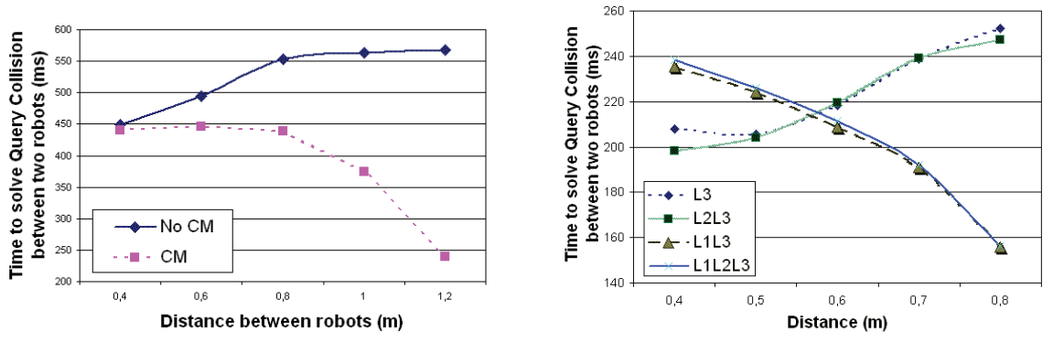

Fig. 3. Time necessary to solve a collision query between two Staubli RX60B robots.
a) Comparison between using the Collision Matrix (CM) or not (NoCM)
b) Comparison using different resolution levels. L3: Geometrical model of each link. L2: The OBB of each link. L1: Bounding box of the robot.

Each one of the proposed optimizations improves the performance of the original library, PQP. However, the combined use of all of them improves even more the global performance of the application. First of all, a query collision between the whole bounding boxes of both robots is performed. If at this level the collision cannot be solved then it is necessary to study collisions among the whole set of links of both robots. The order in which these queries must be performed is given by the Weight Matrix. The query finishes as soon as a collision appears between a pair of links either in the second or third level, or when all pairs have not reported any collision. For each pair of links, the second and third representation levels are studied consecutively, so if a collision is detected in the second level, the third level has to be studied as well.

4. Applications

One of the advantages of making RPQ generic (although it is optimized for robots) is that this library can be applied in a wide set of applications. For instance, in Fig. 4. a robotic aid for laparoscopic surgery is shown. In this application the robot manipulates a laparoscope attached at the end effector. In this section, two specific applications are described in detail. First one has the aim of helping the surgeon to make a cut on a rigid tissue, for example in the cranium, avoiding undesired collisions between the patient and the driller. The surgeon

guides freely the driller that is held by the robot acting in a passive mode, allowing all movements except those which produce undesired collisions. The second application describes a multirobot cooperation system than allows the dynamic expansion of the working space.

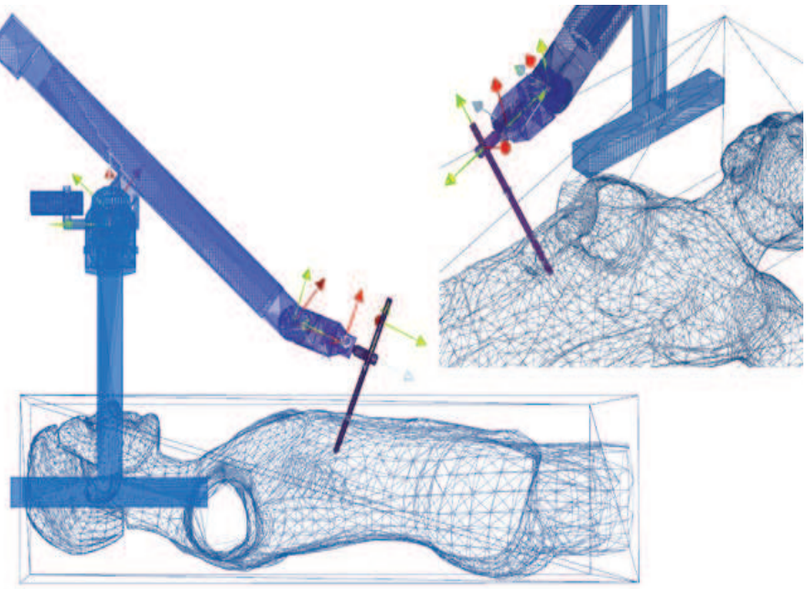

Fig. 4. RPQ library applied in robotic aids for laparoscopic surgery procedures.

4.1 Virtual Fixture and Surface Navigation

Virtual fixtures are constraints that rule the behaviour of the robot that are specifically designed to prevent motion into a forbidden region of the workspace. In this work, the surgeon guides freely a driller held by the slave robot. The main idea is to develop a system helpful for the surgeon that prevents undesired drillings.

Strategy Description

The main goal of the system is to give the robot a reactive behaviour by means of the definition of virtual objects in the PQP Scenario with two objectives:

- Protect the patient from the robot and the driller.
- Help the surgeon to locate the desired cutting area.

PQP´s ability to check collisions in real time allows us not only to achieve these objectives but to operate in an efficient manner.

Throughout a simple interface, in the pre-operative phase, the surgeon specifies the exact location and shape of the cut that must be done. With that information, the system generates a shield that covers the patient while it adapts perfectly to the target area defined by the surgeon, as shown in Fig. 5.

The system gives the surgeon the confidence of knowing that the robot will never let him approach the patient in a non-desired area. However, while the surgeon does not attempt to cross the shield, the robot can be moved freely.

Fig. 5. Detail of a virtual shield and the scenario including the robot arm.

The library developed is useful not only to avoid collisions but also to navigate over an object's surface. This surface navigation allows the surgeon to feel smooth movements of the robot when the tool is in contact with the virtual fixtures. The navigation algorithm helps the surgeon not only avoiding the forbidden regions defined in the pre-operative phase but also guiding him to the desired cutting path.

The navigation algorithm modifies the position of the robot tool, but not its orientation. The algorithm is based on three steps: Knowing the new desired destination of the robot tool, the first step consists of detecting all collisions between the tool and the object's surface. When all collisions are detected, the second step consists of projecting the desired point to the plane of the collision triangle, Fig. 6.a. Finally the projected point that is closer to the desired one is selected. A problem can occur when the projected point falls outside the triangle region. In this situation it is not possible to ensure that this new projected point is always outside the object Fig. 6.c. In this case the new point is projected to the perimeter of the triangle. To accomplish this, the outside region of the triangle is divided into six new regions ($R1, R12, R2, R23, R3, R31$), which are defined by the normals of the edges of the triangle applied to the vertices of the triangle. The point is then projected to the closest point of the triangle border of its region Fig. 6.b. Now, the new destination point is collision free Fig. 6.d.

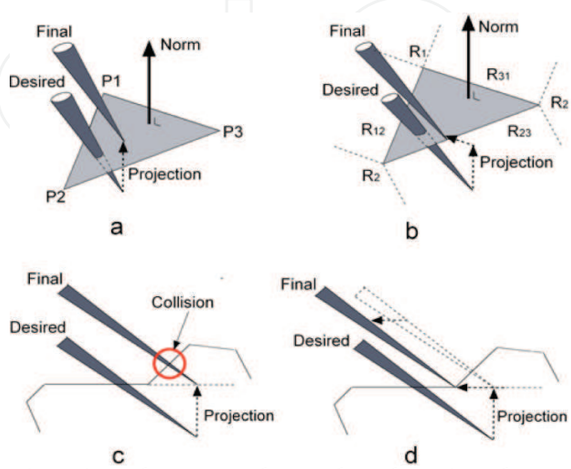

Fig. 6. Different contact situations between the tool and a triangle or set of triangles

4.2 Virtual robot

The concept of the Virtual Robot

In this section, an application that makes an intensive use of the proximity queries package RPQ is presented. During the last years, robotic applications are expected to solve new and more complex tasks. These greater demands requiring more dexterity and accessibility to larger and more complex working areas may imply that a single robot is not enough to perform the desired task. Let's consider as an example of application the need to remotely manipulate an object in front of a camera, for its inspection, in a complex workspace by means of teleoperation. The visualization of the object from all the desired points of view may require a great accessibility and manoeuvring. This manipulation may present some problems like singularities, joint limits, obstacles, or even occlusions produced by the own robotic arm. To deal with such applications the proposed solution is a virtual robot, constituted by a combination of a given number of robots acting as only one. These robots transfer the inspected object from one to another only when necessary, allowing the user to execute the desired task. To follow a continuous trajectory, at least one robotic arm must hold the inspected object at a time. When the user moves the inspected object at will, the robot that holds the object can fall into a singularity or collide with an obstacle. If this happens, the task cannot be accomplished. To prevent this situation, another robot should grasp the object and continue the trajectory before the previous one reaches the critical situation.

The virtual robot is expected to act as if there was only one physical robot, operating in such a way that its internal mode of cooperative working is transparent to the user. To accomplish this goal, the virtual robot has to automatically decide which real robot, or robots, is operative at each successive step during the execution of a task. The virtual robot also has to decide the best configuration for each real robot in order to facilitate the task transfer if necessary and, what is even more important, avoid collisions. In summary, the virtual robot is a set of real robot arms automatically controlled acting as if they were only one, letting the user to concentrate its efforts in the task.

Obviously, two or more cooperative robots can increase the range of tasks that a single robot can execute. In the other hand, the risk of collision between them and with other possible obstacles in the workspace increases respect to work with a single robot.

Collision avoidance in Cooperative Robots

When two or more robots share the workspace, the collision avoidance is done depending on if the trajectory is planned or not. If the trajectory is defined a priori, the path planning algorithm must deal with the collision problem. There is a wide range of different solutions in the literature. In (LaValle, 2006) a detailed study of the most important path planning algorithms is done. In those cases, the time spend on detect collisions is important, but not critical. In the other hand, when the trajectory is not predefined, the collision detection must be done during the execution time. In this case, the computational performance of the collision detection algorithm is critical. In summary, in both cases, but even more in the second one, a fast proximity queries package is necessary.

Most of the proposed solutions in real time collision detection use reactive behavior algorithms. In (J. TaeSeok et al., 2004) a virtual impedance method using a virtual mass-spring-damper model is proposed to avoid collisions in a teleoperated mobile robot. This method is also used as virtual guidance to the robot to achieve the desired end point of the trajectory. This virtual model is applied between the obstacles and the robot and its proportional to the distance between them and the approximation velocity. This principle is

easy to extend to robotic arms working in a six degrees of freedom workspace. In Fig. 7.a an example of the virtual mass-spring-damper model applied to a couple of robots is shown. In this example two virtual restrictions are applied: one between the end effectors of the robots and the second one between the robotic arms. Another technique to avoid collisions and similar to the concept explained above is based on applying a virtual repulsion force field. For instance, in (Jih-Gau Juang, 2004) a virtual repulsion force is proposed to avoid collisions. In this work, the force is proportional to the distance between the robot and the obstacle. This virtual force field can be applied to the whole robot arm, as shown if Fig. 7.b. All of those methods require the knowledge of the geometrical model of the robotic arms and the objects in the workspace. They also need sensorial information of the workspace and the configuration of the robots at each instant of time. Using this information and with a real time proximity query package the collision avoidance algorithms can work properly.

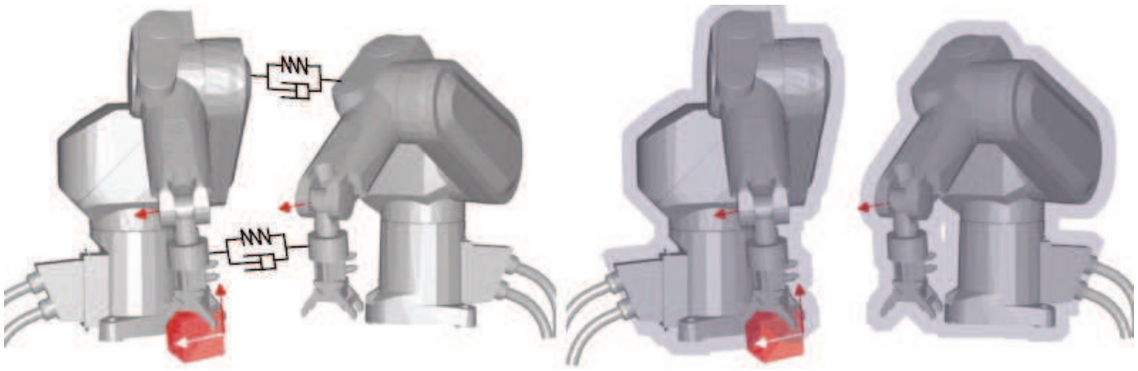

Fig. 7.a. Two virtual mass-spring-damper model are applied to two robotic arms
Fig. 7.b. A repulsive force field, illustrated as an aureole recovering each arm, is applied to the robots in order to avoid collisions.

Applied Algorithm to prevent collisions in Virtual Robot.
The virtual robot is an excellent example of the need and usage of a fast and accurate proximity query package like RPQ. The first reason is, as explained above, that the virtual robot is composed by a set of real robotic arms sharing their workspace, so it implies high risk of collision. Second reason is the absence of path planning, which implies that the collisions must be detected in real time.
Another important reason to use RPQ is because the virtual robot tries to alter the user's desired trajectory as less as possible. To accomplish it, the robot arm that is executing the task does not modify its trajectory to avoid collisions. The rest of the robots must change, if necessary, their configuration to evade collisions and also to facilitate the task transfer. In order to prevent collisions between the robotic arms, at each step of the task the minimum distance between the robot that executes the task and the rest of the robots is calculated. If this distance is lower than a boundary distance, a repulsion force is applied to avoid collision. The boundary distance is defined by the difference between the approximation velocity between the robots and the maximum acceleration and velocity allowed to each robotic arm in the direction of the minimum distance vector. This method implies to calculate at each step of the trajectory the minimum distance vector between the robotic arms. Finally, to improve the time to make the task transfer, the manipulated object has an

attraction virtual force which is applied to the end effector of the robots that does not execute the task. With this force the end effector are near the object. In order to ensure the safety, the attractive force is not applied if it is in conflict with the repulsive force. The combined use of repulsion force, F_{rep}, around the robotic arm and the attraction force, F_{atr}, around the manipulated object is shown in Fig. 8.

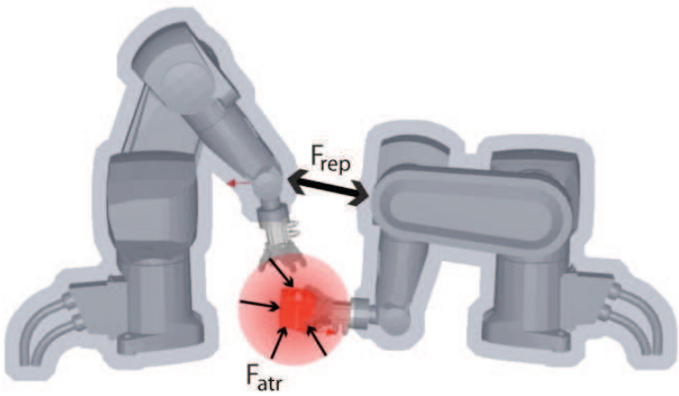

Fig. 8. Application of virtual forces in a robotic cooperative tasc

5. Conclusion

This paper presents the development of RPQ, a proximity queries library optimized for applications where robots share a common workspace and interact with objects. Due to the amount of collision detection packages, RPQ is built above a generic collision package, PQP. It is the generic collision package that better fits RPQ purposes.

The goal of developing RPQ was to fill an existing gap in computer tools for robotic applications, where robots interact with objects in their environment. Three optimizations have been performed to a generic collision library: working with different resolution levels of representation, the use of a weighted matrix for choosing the best order for collision checking and the definition of a binary matrix that determines the possibility of collision between objects. RQP has been validated in different applications such as multirobot collision avoidance, virtual robot controller and surface navigation.

As expected, optimizations improve the time performance of the system, although this improvement is highly application dependent.

The introduction of different levels of resolution in the geometric models of the objects and robots generally decreases the computational time for collision checking. The use of bounding boxes decreases drastically the number of high resolution queries needed. This is a really important point taking into account that they are much more time consuming. There are cases where low resolution queries do not solve the whole collision query. This increases the computation time. However, choosing a suitable order for checking collisions helps finding them in a quicker manner. The precomputation of impossibilities of collision between different objects (Collision Matrix) increases the performance of the system in case of having objects with restricted mobility in the workspace.

The combined use of optimizations generates good results in workspaces shared by at least two robots and objects.

RPQ has a wide range of applicability. RQP library is not only useful for proximity queries but has also proved to be a good tool for surface navigation and virtual representations, due to its ability to introduce virtual objects in the shared workspace. The virtual fixtures developed in the paper are examples of how RPQ can be used to modify robot's behaviour.
As proved in the applications presented, RPQ is not only useful for developers of robotic applications, but also for users of robotic applications, i.e. surgeons that require new robotic tools for improving surgical procedures.

6. References

Bergen, G. V. D. 2002. Solid collision detection library users guide.

B.Geiger (2000). Real-time collision detection and response for complex environments. *International Conference on Computer Graphics*. IEEE Computer Society.

Giralt, X. and Hernansanz, A. (2006). Optimization of proximity queries in robotic environments. *AVR - 2es Jornades UPC de Recerca en Automtica, Visio I Robotica* (in catalan).

Kim, Y., Lin, M., and Manocha, D. (2002). Deep: Dualspace expansion for estimating penetration depth between convex polytopes. *International Conference on Robotics and Automation. IEEE*.

S.Ehmann and Lin, M. (2000). Accelerated proximity queries between convex polyhedra by multilevel voronoi marching. *Technical report, Department of Computer Science*, University of North Carolina.

S.Ehmann and Lin, M. (2001). Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Eurographics*, volume 3.

Stanisic, Z., Jackson, E., and Payandeh, S. (1996). Virtual fixtures as an aid for teleoperation. *9th Canadian Aeronautics and Space Institute Conference*.

UNC. (1999). Pqp - a proximity query package by research group on modeling, physically-based simulation and applications.

La Valle. (2006). Planning *Algorithms Cambridge University Press*.

J. TaeSeok, L. JangMyung, and H. Hashimoto. (2004). Internet-based obstacle avoidance of mobile robot using a force-reflection. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* , 4 ed 2004, pp. 3418-3423.

Jih-Gau Juang. (2004). Repulsive force control based on distance computation algorithm. *Computers and Industrial Engineering*, vol. 47, no. 1, pp. 45-60, Aug.2004.

New Developments in Robotics Automation and Control

Edited by Aleksandar Lazinica

ISBN 978-953-7619-20-6
Hard cover, 450 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

This book represents the contributions of the top researchers in the field of robotics, automation and control and will serve as a valuable tool for professionals in these interdisciplinary fields. It consists of 25 chapter that introduce both basic research and advanced developments covering the topics such as kinematics, dynamic analysis, accuracy, optimization design, modelling , simulation and control. Without a doubt, the book covers a great deal of recent research, and as such it works as a valuable source for researchers interested in the involved subjects.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Xavier Giralt, Albert Hernansanz, Alberto Rodriguez and Josep Amat (2008). Robotic Proximity Queries Library for Online Motion Planning Applications, New Developments in Robotics Automation and Control, Aleksandar Lazinica (Ed.), ISBN: 978-953-7619-20-6, InTech, Available from: http://www.intechopen.com/books/new_developments_in_robotics_automation_and_control/robotic_proximity_queries_library_for_online_motion_planning_applications

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen