

# Real-Time Collision Avoidance Algorithm for Robotic Manipulators

Paul Bosscher

Daniel Hedman

Harris Corporation

Government Communication Systems Division

2400 Palm Bay Road, Palm Bay, FL, 32905

[pbossche@harris.com](mailto:pbossche@harris.com) [dhedman@harris.com](mailto:dhedman@harris.com)

(321) 727-6315

## ABSTRACT

Many applications exist where one or more robotic manipulator arms must work in a crowded or cluttered workspace and must avoid colliding with each other or with obstructions within the workspace. Pre-computing collision-free trajectories is only practical in highly-structured environments with unchanging manipulator trajectories. If the environment is dynamically changing or if nearby manipulators are performing unplanned motions, an on-the-fly adjustment to the manipulator motion is necessary. This paper presents a method for calculating collision-free motion of manipulator arms in real time. This method is based on a reactive paradigm, where no *a priori* knowledge is assumed to be known of the motion of the objects/manipulators to be avoided.

Because of the computational demands associated with performing real-time collision avoidance existing methods for collision avoidance often rely on very coarse methods, including keep-out zones and occupancy grids. While the simplicity of these methods enables real-time implementation, they often overly limit the manipulator motion. The objective of this work is to create a method for performing real-time collision avoidance that does not impose unnecessary limitations on manipulator motion, but remains computationally efficient enough for practical implementation.

When considering collision-free motion of a manipulator, it is necessary to ensure the robot does not collide with itself, other manipulators, or objects in the environment, and that any motion commanded to the manipulator should not violate its joint angle limits or joint velocity limits. The method described in this paper creates a set of constraints on the allowable robot motion based on these limits. These constraints are combined with the desired motion of the manipulator within a constrained optimization framework. This optimization that can be calculated very efficiently which allows collision-free motions to be computed in real time.

This paper also presents experimental results for the implementation of this algorithm on a pair of industrial manipulators. The two manipulators (an Epson PS5 ProSix and a Mitsubishi PA-10C) are six-axis anthropomorphic arms. In this implementation we chose to use a master-slave approach, where the master robot is given priority and may execute any arbitrary motion path. The slave robot is commanded to attain a desired motion but it may deviate from this path in order to avoid collision with the master robot. Control of the slave robot was implemented on a PC running Windows XP. The control software was implemented in the Matlab Simulink environment. Using Quanser QuaRC software the Simulink models run real-time using the timing functionality of a Quanser QuaRC Q8 hardware-in-the-loop board. Custom Simulink blocks written by Quanser are used to interface with the Mitsubishi (slave) manipulator. In order to avoid the master (Epson) robot the collision avoidance algorithm needs joint angle data from the Epson robot; thus joint encoder data is sent to the collision-avoidance controller via an Ethernet connection to the Epson controller.

Experimental results have demonstrated that our proposed collision avoidance algorithm successfully prevents collisions between the two manipulators at a variety of speeds. Moreover, the specified minimum safe distance is never violated. Results also demonstrate that the algorithm can easily handle many simultaneous constraints without significant increase in processor burden, enabling the algorithm to be implemented in environments with many potential simultaneous collisions that must be avoided.

A second experimental setup was also created in which an object in the workspace (a ball on a stick) could be tracked using a machine vision system. Using a 1024 x 768 camera images of the workspace are captured and processed (using a GPU) at a rate of 20 Hz to determine the position of the ball in the workspace. This data is sent to the collision avoidance controller and based on the position feedback obtained from the vision system the robot is able to successfully avoid collision with the object. This demonstrates that the collision avoidance algorithm can successfully operate for varied objects sensed via diverse sensing methods. This experiment also illustrates that the algorithm's effectiveness is bounded by the bandwidth of object position feedback, as very fast motions of the ball toward the robot could not be avoided due to delay in sensing the position of the ball.

Future work on this algorithm includes integration with a global planner, which could provide additional trajectory modification based on predicted motion of objects in the robot's workspace.