POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Real Time Distributed Approach to Collision Avoidance for Industrial Manipulators

*Original*
A Real Time Distributed Approach to Collision Avoidance for Industrial Manipulators / Fenucci, A., Indri, M., Romanelli, F.. - ELETTRONICO. - (2014). (2014 IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2014) Barcelona (Spain) 16-19 Settembre 2014).

*Availability:*
This version is available at: 11583/2565550 since:

*Publisher:*
IEEE - INST ELECTRICAL ELECTRONICS ENGINEERS INC

*Published*
DOI:

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

(Article begins on next page)

14 November 2026

# A Real Time Distributed Approach to Collision Avoidance for Industrial Manipulators

Alba Fenucci, Marina Indri
Dipartimento di Automatica e Informatica,
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
alba.fenucci@studenti.polito.it
marina.indri@polito.it

Fabrizio Romanelli
Research & Development group in Motion and Control of Comau Robotics
Via Rivalta 30, 10095 Grugliasco, Italy
fabrizio.romanelli@comau.com

## Abstract

*Robot interaction with the surrounding environment is an important and newsworthy problem in the context of industrial and service robotics. Collision avoidance gives the robot the ability to avoid contacts with objects around it, but most of the industrial controls implementing collision avoidance checks only the robot Tool Center Point (TCP) over the objects in the cell, without taking into account the shape of the tool, mounted on the robot flange. In this paper a novel approach is proposed, based on an accurate 3D simulation of the robotic cell. A distributed real time computing approach has been chosen to avoid any overloading of the robot controller. The simulator and the client application are implemented in a personal computer, connected via a TCP-IP socket to the robot controller, which hosts and manages the anti-collision policies, based on a proper speed override control. The real time effectiveness of the proposed approach has been confirmed by experimental tests, carried out for a real industrial setup in two different scenarios.*

## 1. Introduction

Industrial robots give an important contribution to the competitiveness of modern industries. They are asked to accomplish their tasks in a workspace containing obstacles as fast as possible or with minimal energy consumption, while preserving a strong autonomy. The capability of managing changes in the environment [17] would represent an important step toward a real autonomy of the robot behavior. Different technological challenges somehow limited until now the robots functionality outside rigidly structured environments. In the real industrial context the robot is asked to perform complex tasks while moving in a crowded environment. In most of the cases there are both static and dynamic obstacles [18], such as other robots or other moving parts inside the cell, as depicted in Figure 1.

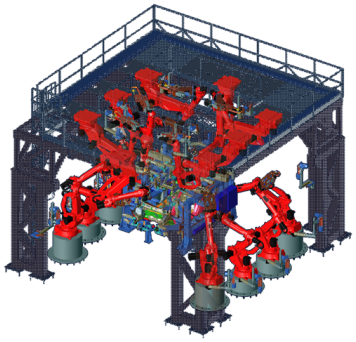


Figure 1: Example of a crowded industrial environment.

The interested reader can find details about the current state of the art of collision avoidance in industrial contexts in literature, e.g., in [6], [7], [10], and [21], and about generic collision avoidance paradigms in [3] and [13]. An interesting solution towards an actual robot-environment interaction has been proposed in [16], where hybrid control techniques for static and dynamic environments have been developed, but with a limited validity, since only the Tool Center Point (TCP) of the robot was considered for collision avoidance, without taking into account its complex geometry and shape. Although this method was applied on a real-time system, it has the drawback of not being suitable for complex industrial applications, which require a control system able to cope with sophisticated shapes.

In this paper an effective approach to collision avoidance for industrial manipulators is developed, starting from an accurate 3D model of the entire robotic cell, in order to formulate a new paradigm able to face the lack of complex geometry management that characterizes other approaches. Several techniques from computer graphics [1], [5], [11], [12], and [19] have been analyzed to compute the minimum distance between 3D objects. In particular two 3D creation/modeling tools seemed to be the most promising ones, Blender [8] and V-REP by Coppelia Robotics [9]. The V-REP platform was finally chosen for different reasons, as discussed in the next sec-

tion, and especially because it includes a versatile Mesh-Mesh distance calculator module. A distributed real time computing approach (see e.g., [20], [22], and [23]) has been chosen to avoid any overloading of the robot controller, implementing the simulator and the client application in a personal computer, connected via a TCP-IP socket to the robot controller, which hosts and manages the anti-collision policies. The proposed procedure, which is mainly based on proper policies of speed override control, can be applied to different manipulators and/or to various configurations of the robotic cell, possibly including more robots simultaneously working inside it. In particular, two scenarios have been considered for the experimental tests: a redundant 7-dof robot prototype moving in presence of a static obstacle that must be bypassed, and two Comau 6-dof industrial manipulators (a SMART5 SIX and a SMART NS12) cooperatively working in a cell.

The paper is organized as follows. An overview of the proposed collision avoidance approach is given in Section 2, while Section 3 is devoted to its software architecture, including the Virtual 3D model, the client application and the robot control unit, hosting the speed override policies for collision avoidance. The effectiveness of the proposed approach is experimentally tested in Section 4 in the two considered scenarios. Section 5 draws some final conclusions and highlights the potentialities of the proposed platform for future developments.

## 2. Overview of the collision avoidance approach

The goal of the proposed approach is to avoid any collision between a robot and other (static or dynamic) objects in the cell, by adequately modifying the robot speed override on the basis of the minimum distance between any part of the robot and any element in the cell. A key point is given by the capability of correctly and efficiently determining such a minimum distance. A possible, simplified solution would consist in modeling the robot, the tool, and the items in the cell by composing simple solids, like spheres, cylinders and boxes. But in this way all the modeled objects would be bigger than the real corresponding ones, and moreover the results thus obtained would be inaccurate, because the solids could not perfectly model all the objects in the scene. The first element of the proposed approach is then given by an accurate virtual replica of the status of the cell, developed in the V-REP simulator, where distances between complex objects in the robotic cell can be correctly computed. V-REP [9], which is generally used for factory automation simulations, allows an accurate modeling of the robot and of the entire cell, including objects of different, complex shapes. It has been chosen for various reasons: it is supported from several programming languages (C/C++, Python, Java, Matlab), it does not require strong computational resources for a complex simulation, it can simulate more robots simultaneously, and it is an open source software with free license for educational purposes, but most of all it includes a Mesh-Mesh distance calculator module that allows fast minimum distance calculations between any *shape* (convex, concave, open, closed, etc.) in the scene.

The architecture of the proposed approach includes, besides the 3D simulator, a client application module and the robot controller (Figure 2).

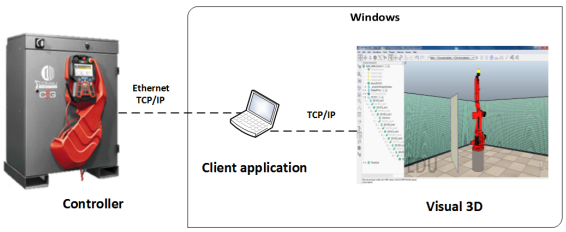



Figure 2: Hardware/software architecture.

The client application is the module that manages the data exchange between the simulator and the unit controller, so to update the virtual robot configuration coherently with the joint position values acquired from the real robot. The robot controller is used for motion planning and control of the robot, the acquisition in absolute reference of the axes positions, and the generation of the references to the drive modules. The controller implements, too, the speed override control policies for collision avoidance. Two different anti-collision policies have been implemented in order to define the so called *warning zone*, which surrounds the robot and is used to detect the risk of a possible collision, and consequently apply the proposed speed override control laws. As discussed in Section 3.3, the width of such a warning zone can be constant or depending on the characteristics of the robot motion.

One of the most peculiar and innovative aspect of the proposed paradigm is its *distributed* processing approach. The simulator and the client application are not implemented in the robot controller, but they run in a separate, external processor, thus avoiding the centralization of all the computational burden into a single entity. For this reason all the calculations relative to the evaluation of the minimum distance from the elements in the cell are executed by the personal computer (having an Intel i7 2.5 GHz processor, a RAM memory of 4 GB and a 500 GB HD), whereas the robot motion and control are managed by the robot controller. In this way the greater computational power of the external PC can be fully exploited, avoiding any risk of overloading the robot controller. Thanks to its distributed structure, the proposed approach can be applied to manage cells including multiple robots, working simultaneously, independently or in collaboration. In addition, the client application allows the user to include a different robot inside the cell quite easily. The flexibility in making the changes is made possible thanks to the previous development of the 3D models of the possible, various robots. The physical connection among the controller and the application is via TCP-IP socket, as well as the connection between the client application and the simulator. At this stage of development

the software client application and the simulator run in a Windows-based not embedded operating system, obtaining good results, as illustrated in the next sections. The development of the client application and of the simulator in a Linux-based system has been developed in a non-real-time context, and a full development for the real-time context is in progress.

## 3. Software architecture

In this section the main software components of the architecture sketched in Figure 2 are detailed: the virtual 3D simulator, the client application, the robot controller unit and the communication protocol. The avoidance collision policies, implemented in the controller, are also described.

### 3.1. Virtual 3D simulator

The main module of the architecture is the simulator, which allows an accurate simulation of the real cell and evaluates at each time instant the minimum distance value between the robot and any other item in the cell. The simulator has a distributed structure, so that each model can be controlled through an embedded script or an API called by a remote client. The simulated scene is made-up of different scripts written in LUA: the base script task is devoted to execute the client application, the other scripts are related to the models of the items introduced into the scene. Each model has its own script that allows it to exchange data with the client application. After the creation of the cell and the beginning of the simulation, the simulator receives the joint position values of each robot as inputs (more than one robot could be present in the cell); while the simulation goes on, the joint values are continuously updated and the virtual robots consequently update their configuration. Using its devoted Mesh-Mesh module, the simulator computes the minimum distance between each robot in the cell and any element inside it, so to determine the overall distance minimum value (i.e., for which part of which robot in the cell a collision would be going to happen). The Cartesian position of the robot point corresponding to the minimum distance situation is returned by the simulator and transmitted to the client application. Thanks to this information, it is possible to know which link of the robot is involved in the upcoming collision, and hence properly modify its speed override, as discussed in Section 3.3.

### 3.2. Client application

The client application is written in C++ using the APIs from V-REP; this is the core of the software architecture and it is responsible for the management of the information exchanged between the Virtual 3D simulator and the C5G controller. In order to accomplish this task, the C++ application is provided with two separate clients: the first one manages the connection with the C5G controller through TCP-IP sockets, the second one establishes a socket communication with V-REP, so to reduce delay and network load to a great extent. The client application is also demanded to find which robot link corresponds to the minimum distance measured and to acquire the velocity signal of this link for all the robots in the cell. The results of all such computations are sent to the controller.

### 3.3. Robot control unit and speed override control

In this approach, the controller has two main tasks: (i) to supply to the simulator the positions of all the joints of each robot in the cell, and (ii) to take decisions about possible collisions.

The exchange of messages between the controller and the application is accomplished through a protocol based on bidirectional communication, where a client sends a message and the server always answers to that message. The exchanged messages, which are treated as ASCII strings, are:

- the joint position request of all the controlled robots;
- for each robot the minimum distance measured from an external object, the corresponding link and its velocity.

The proposed anti-collision policies are based on robot speed override control laws to be applied when the risk of a collision is detected, i.e., when the robot minimum distance $d$ from any obstacle is smaller than a safe threshold value $d_{max}$. Two different anti-collision policies have been formulated, both based on the definition of a proper *warning zone* surrounding the robot, characterized by two limits: the upper bound $d_{max}$ and the lower bound $d_{min}$. The basic idea of the policies is that if $d_{min} < d < d_{max}$, the velocity must be adequately reduced, because otherwise a collision is probably going to happen, whereas if $d < d_{min}$ the robot must be stopped as soon as possible. In the first proposed anti-collision policy the width of the warning zone is constant, i.e., $d_{max}$ and $d_{min}$ are constant, and the speed override is uniformly changed inside it according to the following equation:

$$vo_k = \begin{cases} vo_{k-1} \cdot \dfrac{d - d_{min}}{d_{max} - d_{min}} & \text{if } d_{min} < d < d_{max} \\ vo_{k-1} & \text{if } d \geq d_{max} \\ 0 & \text{if } d \leq d_{min} \end{cases} \quad (1)$$

where $vo_k$ is the speed override of the robot at the $k$-th time instant. If the distance is smaller than the lower bound of the warning zone, the robot immediately changes its state to the HOLD mode, in which its motion is stopped as soon as possible. It must be underlined that only in case of approaching an obstacle a speed change is applied, by checking the link velocity considered in the measurement of the distance. If the robot is moving away from the obstacle, its speed is multiplied by a constant, to avoid any unnecessary slow down. $d_{max}$ is chosen based on the stopping distance of the robot, considering its highest speed and acceleration; on the other hand, $d_{min}$ represents a *cushion* distance in millimeters from the collision area.

A modified version of the speed override control has been proposed considering the width of the warning zone

as a function of the characteristics of the motion of link $i$ involved in the collision risk; in particular in this second policy the upper bound $d_{max}$ varies according to the following equation:

$$d_{max} = \begin{cases} \frac{v_i}{2a_i} + \frac{v_i a_i}{2j_i} & \text{if } d_{max} \geq \bar{d} \\ \bar{d} & \text{if } d_{max} < \bar{d} \end{cases} \quad (2)$$

where $v_i$ is the velocity of the considered link, $a_i$ is its acceleration, $j_i$ is its jerk, and $\bar{d}$ is the minimum value allowed for $d_{max}$ for safety reasons (in order to guarantee a proper stopping space in any case). The speed override reduction has still a linear progression as in (1), but the width of the warning zone varies, and in particular a higher reduction of the robot velocity is imposed, when the link involved in the risk of a collision is moving very quickly. In this way, as the link velocity increases, the warning zone grows in order to take into account that a wider stopping space is needed.

### 3.4. Communication protocol

Three main phases can be distinguished into the communication flow among the architecture components previously described (as sketched in Figure 3): (i) initialization, (ii) collision avoidance management, and (iii) termination.

In the first phase, after the creation of the virtual cell and the simulation start, the main script launches the client application executable. The first task of the C++ application is then the connection to the two servers (i.e., Virtual 3D and C5G Controller).

The second phase is relative to the key point of the whole process, i.e., the collision avoidance procedure. In this phase the client application interrogates the controller about the joint position values of all the robots in the cell; after receiving these data, the application accordingly updates the virtual robots configuration. The Virtual 3D simulator sends out to the client application the minimum distance found between each robot and any element in the environment, and the data relative to the corresponding link. The communication between the Virtual 3D simulator and the client application is based on creation and deletion of signals, using a persistent global buffer. In order to ensure the synchronicity between the application and the simulator, and the update of the data, the simulator looks for the minimum distance and sends out a new message only when all the other signals with the same name are not active. On the other hand, the application deletes the signal when it receives a new joint position from the controller. The signals are created according to the number of robots instantiated on the scene to be managed: this task is performed by each robot model script involved in the scene. The application finally sends to the robot controller the minimum distance value, the link number and its velocity for the application of the collision avoidance policies described in Section 3.3.

The last phase of the communication flow occurs when the simulation ends: the client application closes all the communications to the servers and terminates its execution.

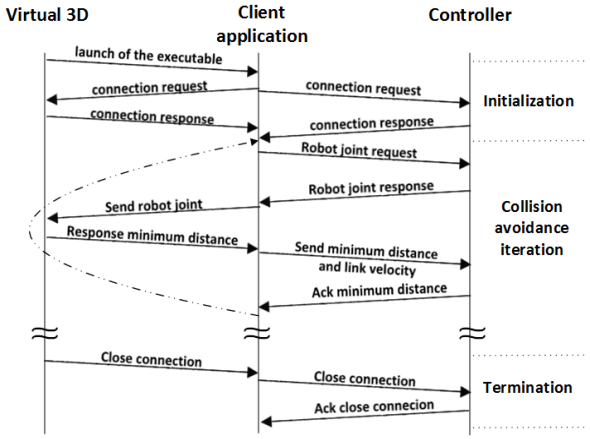



Figure 3: Communication flows.

## 4. Experimental tests

The effectiveness of the proposed approach has been experimentally validated in two different scenarios in the RoboLAB, the new joint Comau-Politecnico di Torino industrial robotics laboratory. The experimental setup, whose structure is sketched in Figure 4, includes different manipulators in the two scenarios, the Comau C5G Controller and a personal computer, hosting the V-REP simulator and the client application. Both scenarios reproduce examples of real welding robot cells: in the first one a redundant 7-dof robot prototype is moving in presence of a static obstacle (i.e., a panel), whereas in the second one two 6-dof industrial manipulators have to move very close to each other, in presence of a static obstacle, too. In both cases the cell has been faithfully reproduced in the virtual 3D simulator, perfectly modeling each involved robots and the grating cell protection. Every element in the scene is placed in the exact position with respect to the real cell, so to have the measured distances in the simulated environment as reliable as possible.

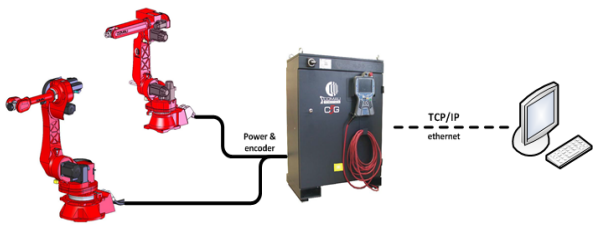



Figure 4: Experimental setup components.

The second element of the experimental setup is the COMAU C5G robot controller (Figure 5). The C5G Control Unit features state-of-the-art processors to control the trajectory and the periphery, the application software and the user's interface, in order to ensure the highest level of

performance of both areas. The processors work through a real time VxWorks operating system. The processor controlling the movements carries out in particular the following functions: (i) application of servo-adaptive algorithms, with dynamic pattern calculated in real time for all the links, based on the load, position speed and inertia conditions; (ii) acceleration/deceleration modulation for joint-like movements, to optimize the robot motor performances in terms of speed; (iii) linear interpolation featuring fly-path total programmability and constant speed; (iv) circular interpolation featuring different orientation evolution possibilities; (v) possibility to control up to 16 axes: (vi) management of interference regions, to control robot TCP against predefined forbidden or monitored zones.

Figure 5: The Comau C5G controller unit.

The last element of the setup is the personal computer, which manages the execution of the virtual 3D simulation and the client application. The control unit is connected with the robots inside the cell (allowing in particular data acquisition from the joint encoders of each robot), and with the personal computer via Ethernet TCP/IP, for the exchange of messages about joint positions and minimum distance values of the robots in the cell. The robots involved in the experimental tests have been programmed using the Comau PDL2 language, as in any standard industrial application.

The proposed anti-collision policies proposed in Section 3.3 (i.e., with the width of the warning zone constant or variable) have been applied to the considered scenarios, imposing the following values to the warning zone parameters:

- $d_{min} = 0.02$ m and $d_{max} = 0.45$ m in the first anti-collision policy (1)
- $\bar{d} = 0.3$ m as minimum value allowed for $d_{max}$ in (2) in the second anti-collision policy.

A preliminary test was performed, setting the robot maximum speed override, to evaluate the communication and computational times required for each obstacle avoidance cycle, i.e., the time interval between the dispatch of the robot joint request from the client application and the acknowledgement of the minimum distance information sent back by the controller. This cycle (sketched in Figure 6) can be divided into three main phases:

- communication/transmission between the client application and the controller (robot position update)
- computation of the collision avoidance algorithms
- communication/transmission between the client application and the controller (robot override modification).

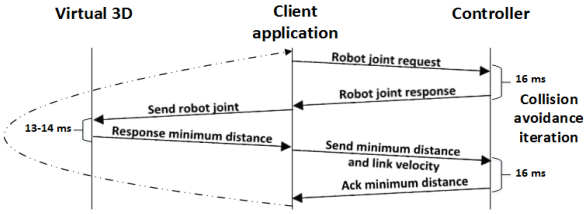



Figure 6: Communication and computational times.

The first phase lasts 16 ms on average, considering the time interval between the robot joint request and the robot joint response. The computation phase lasts about 13-14 ms on average, including also the exchange of messages between the client application and the virtual 3D simulator. The last phase of the cycle, i.e., the time interval between the dispatch of minimum distance and link velocity to the controller and the client application receiving back the acknowledgement from the controller, lasts 16 ms on average.

The tests performed in each scenario are detailed in the next subsections, where the achieved results are reported and discussed.

### 4.1 First scenario: presence of a static obstacle

In the first scenario a redundant 7-dof robot prototype, with a maximum load of 7 kg, has to move near/around a static obstacle, constituted by a panel of size $1.47 \times 1.87$ m, located at a distance of 0.45 m from the robot base. Figure 7 shows the real cell and the simulated one in the virtual 3D environment in this scenario.

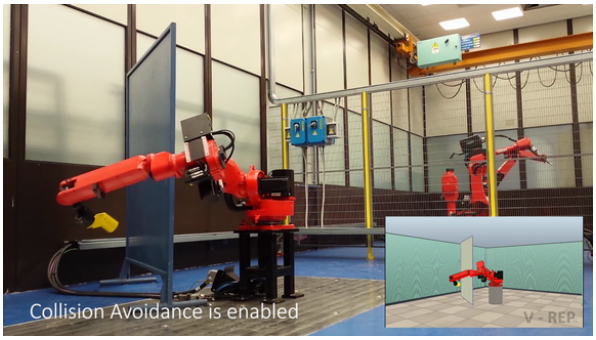
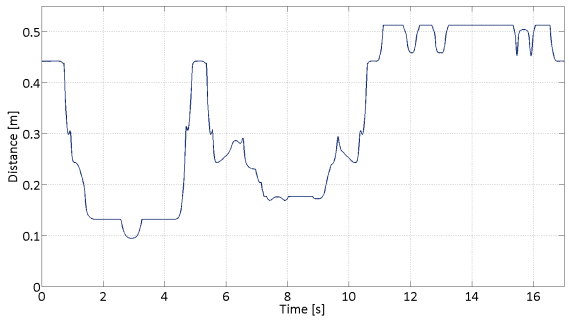


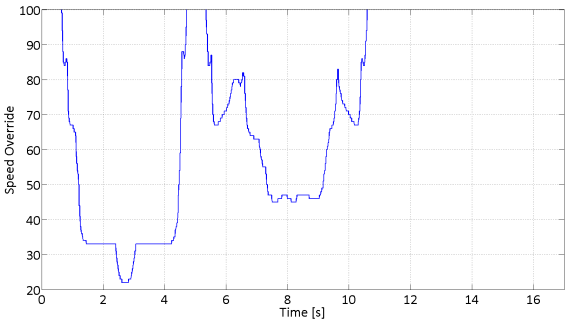

Figure 7: Scenario with the 7-dof manipulator and a static obstacle.

The complete video of the test ("Collision avoidance approach for industrial manipulators: tests with a 7-dof redundant robot and a static obstacle") is available

in the "Video" section ("Industrial robotics" subsection) of the website of the Robotics Research Group (RRG) of Politecnico di Torino (`http://www.polito.it/labrob`). Figure 8(a) reports the time-history of the minimum distance detected between the robot and the panel during the execution of the assigned motion, whereas Figure 8(b) shows the corresponding time-history of the robot speed override. Comparing the figures, it is possible to note the saturation of the speed override to 100% (i.e., no reduction is applied to the planned velocity), when the minimum distance detected is greater than the imposed value of $d_{max}$ (0.45 m).



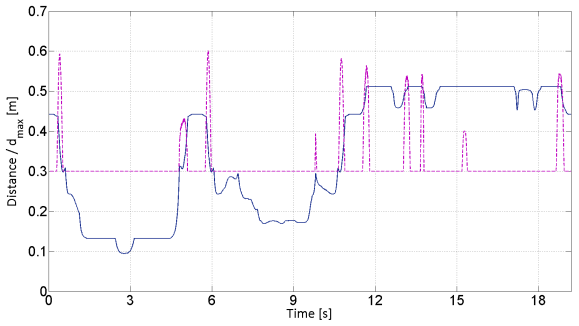

(a) Minimum distance robot-obstacle detected



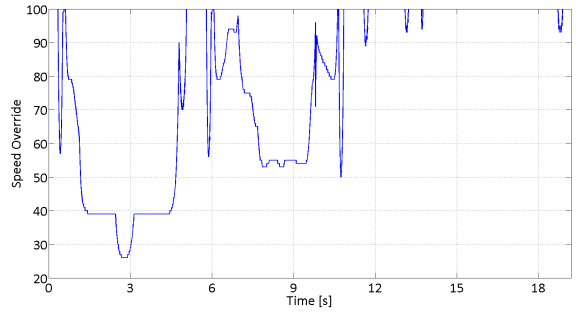

(b) Robot speed override applied

Figure 8: Tests in the first scenario: experimental results with the first anti-collision policy.

When the second anti-collision policy is applied (Figures 9(a) and (b)), if the robot link speed and acceleration are high, the warning zone increases its dimensions according to (2) up to the limit value $d_{max}$. When the warning zone reaches this value, the algorithm acts exactly as in the first anti-collision policy. If the minimum distance from the obstacle falls inside the warning zone, the robot speed override is decreased according to (1), as depicted in Figure 9(b). Comparing Figures 8 and 9, the different behavior of the two anti-collision policies can be appreciated: in the last part of the robot motion for time $t > 11$ s, the second policy seems to be more conservative, since the speed override value is sometimes smaller than 100% in Figure 9(b), but actually during the first part of the motion, when the robot is closer to the obstacle, a smaller velocity reduction is allowed by this policy (compare the speed override values in Figures 8(b) and 9(b) in the time




(a) Minimum distance robot-obstacle (blue) and variation of the upper limit $d_{max}$ of the warning zone (magenta).




(b) Robot speed override applied

Figure 9: Tests in the first scenario: experimental results with the second anti-collision policy.

interval between 1 and 5 s).

### 4.2 Second scenario: a multi-arm cell

The second scenario is relative to a multi-arm cell with two Comau 6-dof industrial manipulators, a SMART5 SIX and a SMART5 NS12, with a maximum load at wrist of 6 and 12 kg, respectively, and high repeatability (0.05 mm). In the carried out tests they have to move very close to each other and near a static obstacle, constituted by a robot wrist, having size $0.25 \times 0.39 \times 1.5$ m, and located at 0.64 m from the SMART NS12 base and at 0.41 m from the SMART SIX base. The real cell and the simulated one in the virtual 3D environment are shown in Figure 10.

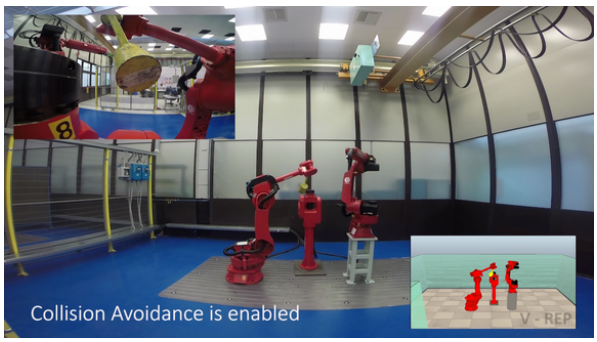
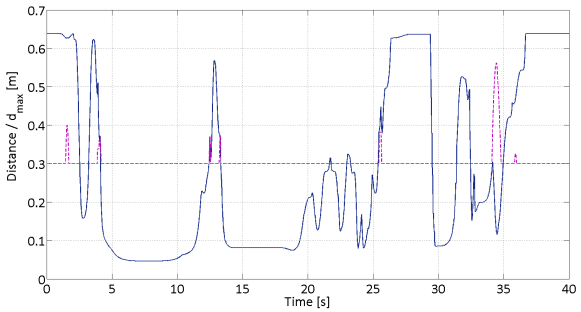
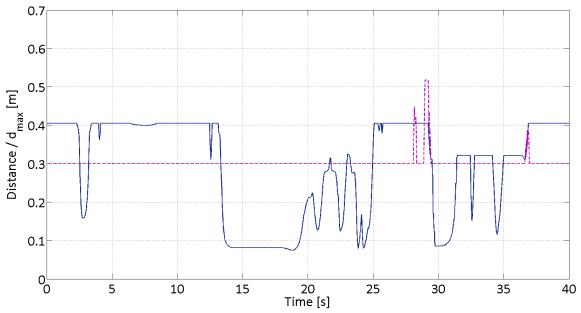



Figure 10: Scenario with the multi-arm cell including two robots and a static obstacle.

In this scenario, in which multiple risks of collision are present (between each robot and the obstacle, and between the robots themselves), the second, more effective anti-collision policy has been directly applied. The complete video of the the test ("Collision avoidance approach for industrial manipulators: tests in a multi-arm cell with two robots and a static obstacle") is available also in this case in the RRG website. Figures 11(a) and (b) show the time-history of the minimum distance detected between each robot and *any* obstacle in the cell, where the closest obstacle during the motion can be given by the static element in the cell (i.e., the robot wrist) as well as the other manipulator. The corresponding time-history of the upper limit $d_{max}$ of the warning zone of each robot is also reported.




(a) Minimum distance Robot 1-obstacles (blue) and variation of the upper limit $d_{max}$ of its warning zone (magenta).

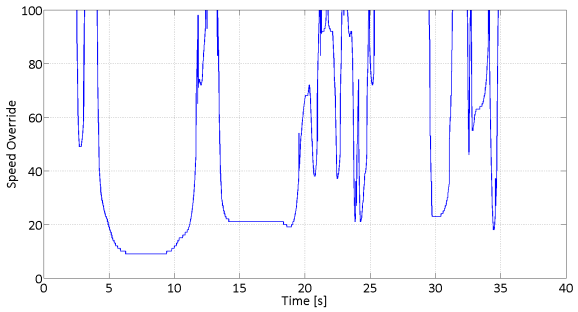



(b) Minimum distance Robot 2-obstacles (blue) and variation of the upper limit $d_{max}$ of its warning zone (magenta).

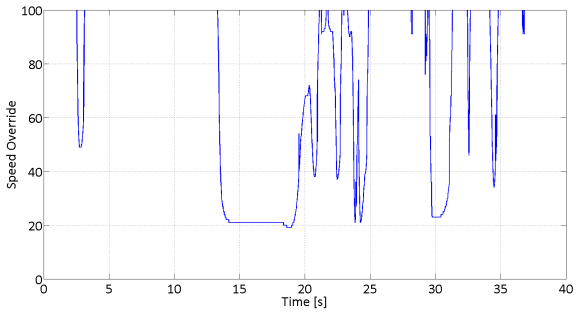
Figure 11: Tests in the second scenario: experimental results with the second anti-collision policy.

In the time interval between 13 and 25 s the trends of the minimum distance detected for the two robots are similar, because they are moving very close to each other, so that the minimum distance condition is referred to the other robot. On the other hand, the two warning zones have different width in the same interval, because the value of $d_{max}$ depends on the motion characteristics (velocity, acceleration, jerk) of the specific link involved in the minimum distance condition.

The efficiency of the applied policy is clearly shown in Figures 12(a) and (b), where the behavior of the speed override applied to each robot is reported: a rapid override increase is applied whenever possible to let each robot move fast away from the warning zone.




(a) Speed override applied to Robot 1




(b) Speed override applied to Robot 2

Figure 12: Tests in the second scenario: experimental results with the second anti-collision policy.

## 5. Conclusions and future works

A fast and distributed method for collision avoidance between robots and obstacles has been proposed in this paper. The core of the approach is a new strategy to evaluate the distance between the robot and any obstacle in its workspace, based on the accurate simulation of the real cell. The computed distance is used to move the robot at a reduced speed in case of an imminent collision. The reported results of the carried out experimental tests have confirmed the real-time effectiveness and good performances of the method. The proposed approach is specifically oriented to the industrial application. In fact, after the construction of the virtual replica of the cell (using available libraries of the robot models previously developed), the proposed paradigm can be applied achieving excellent results in the manual programming mode, which is used to execute movements requiring closeness to the robot (e.g., point teaching, program verification or troubleshooting operations). The approach makes such operations safer, as it allows to prevent collisions between the robot and the obstacles even if the movements are not pre-programmed as in the manual mode, in which the manipulator can be operated only via the teach pendant.

Future works are devoted to the integration of the proposed approach into the Comau C5G Open Controller [2]. This innovative control architecture allows the easy and

safe integration of an industrial robot with an external common personal computer to enable access to the robot control and interaction at different levels in the machine control. The C5G Open Controller is the basis platform for the development of prototypes and to test the developed applications on a fast real-time system. Moreover the C5G Open architecture is particularly suited for hosting the ROS module in order to open the control also to the ROS compliant sensors, drivers and other robotic platforms as reported in [14] and [15], as shown in Figure 13.

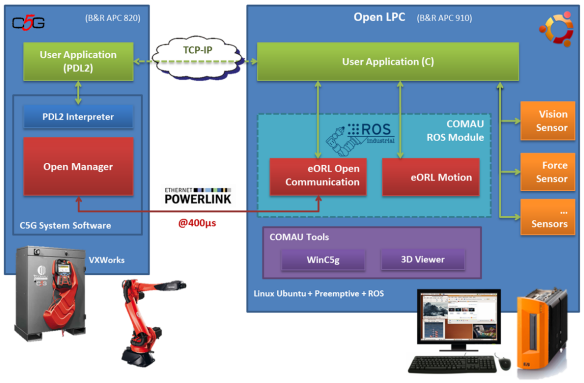



Figure 13: The C5G Open Controller platform.

The very fast communication allowed (up to 400 $\mu$s) thanks to the openPOWERLINK real-time communication protocol and the real-time feature of the operating system (Linux real-time preemptive patch), makes such a controller an ideal platform to develop real-time applications, as required for the full industrial implementation of the proposed anti-collision approach.

## References


[1] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-Time Rendering*. A K Peters, Ltd, 2008.

[2] G. Antonelli, S. Chiaverini, V. Perna, and F. Romanelli. A modular and task-oriented architecture for open control system: the evolution of C5G open towards high leve programming. In *IEEE 2010 Int. Conf. on Robotics and Automation*, Anchorage, Alaska, May 3 2010. Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications.

[3] A. A. Ata and T. R. Myo. Collision-free trajectory planning for manipulators using generalized pattern search. *International Journal of Simulation Modelling (IJSIMM)*, 5:145–154, December 2006.

[4] J. Baumgartner and S. Schoegger. POWERLINK and real-time linux: A perfect match for highest performance in real applications. *Twelfth Real-Time Linux Workshop*, October 25-27 2010.

[5] W. J. Bouma and G. Vanecek. Collision detection and analysis in a physically based simulation. Technical Report 91-055, Computer Science Technical Reports, Purdue University, USA, 1991.

[6] R. A. Brooks. Planning collision free motions for pick and place operations. *Massachusetts Institute of Technology - Artificial Intelligence Laboratory*, May 1983.

[7] A. De Luca and F. Flacco. Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration. In *The Fourth IEEE RAS/EMBS*, Roma, Italy, June 24-27 2012. International Conference on Biomedical Robotics and Biomechatronics.

[8] G. C. Fisher. *Blender 3D Basics*. Packt publishing, June 2012.

[9] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 51–62, Darmstadt, Germany, November 15-18 2010. Second International Conference SIMPAR 2010.

[10] M. Gerdts. Path planning and collision avoidance for robots. *Numerical algebra, Control and Optimization*, 2(3):437–463, September 2012.

[11] E. G. Gilbert, D. W. Johnson, and S. S. Keerth. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2), April 1988.

[12] D. Knott and D. K. Pai. Collision and interference detection in real-time using graphics hardware. *Proceedings of Graphics Interface 2003*, pages 73–80, May 2003.

[13] I. Memon, F. A. Mangi, and D. A. Jamro. Collision avoidance of intelligent service robot for industrial security system. *IJCSI International Journal of Computer Science Issues*, 10(3), March 2013.

[14] M. Quigley, B. Gerkey, K. Conley, K. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *IEEE Int. Conf. on Robotics and Automation*, volume 3, Kobe, Japan, May 12-17 2009. ICRA Workshop on open source software.

[15] B. Rhoden, K. Klues, A. Waterman, D. Zhu, and E. Brewer. ROS: A scalable operating system for parallel applications on many-core architectures. *Research supported by Microsoft Award #024263 and Intel Award #024894*.

[16] F. Romanelli. Hybrid control techniques for static and dynamic environments: a step towards robot-environment interaction. In *Robot Manipulators New Achievements*. Aleksandar Lazinica and Hiroyuki Kawai (Ed.), InTech, 2010.

[17] F. Romanelli. Advanced methods for robot-environment interaction towards an industrial robot aware of its volume. *Journal of Robotics*, 2011:12, 2011.

[18] F. Romanelli and F. Tampalini. A control algorithm for the management of multiple dynamical geometrical areas for industrial manipulators. In *Proc. of the RAAD 2008*. 17th Int. Workshop in Alpe-Adria-Danube Region, September 15-17 2008.

[19] J. O. Rourke. *Computational Geometry in C*. Cambridge University Press, October 13 1998.

[20] D. C. Schmidt, A. Gokhale, R. E. Schantz, and J. P. Loyall. Middleware r&d challenges for distributed real-time and embedded systems. *SIGBED Rev.*, pages 6–12, April 2004.

[21] C. A. Shaffer and G. M. Herb. A real-time robot arm collision avoidance system. *IEEE Transactions on Robotics and Automation*, 8(2), April 1992.

[22] J. A. Stankovic. Real-time and embedded systems. *ACM Computing Surveys*, 28(1), March 1996.

[23] O. Wulf, J. Kiszka, and B. Wagner. A compact software framework for distributed real-time computing. *Fifth Real-Time Linux Workshop*, November 09-11 2003.