Collision Avoidance System with Obstacles and Humans to Collaborative Robots Arms Based on RGB-D Data

Thadeu Brito[1(✉)], José Lima[1,2], Pedro Costa[2,3], Vicente Matellán[4], and João Braun[5]

[1] Research Centre in Digitalization and Intelligent Robotics and IPB, Bragança, Portugal
{brito,jllima}@ipb.pt
[2] INESC TEC - INESC Technology and Science, Porto, Portugal
[3] Faculty of Engineering of University of Porto, Porto, Portugal
pedrogc@fe.up.pt
[4] Research Institute on Applied Sciences in Cybersecurity, León, Spain
vicente.matellan@unileon.es
[5] Federal University of Technology - Paraná, Toledo, Brazil
jneto.1996@alunos.utfpr.edu.br

Abstract. The collaboration between humans and machines, where humans can share the same work environment without safety equipment due to the collision avoidance characteristic is one of the research topics for the Industry 4.0. This work proposes a system that acquires the space of the environment through an RGB-Depth sensor, verifies the free spaces in the created Point Cloud and executes the trajectory of the collaborative manipulator avoiding collisions. It is demonstrated a simulated environment before the system in real situations, in which the movements of pick-and-place tasks are defined, diverting from virtual obstacles with the RGB-Depth sensor. It is possible to apply this system in real situations with obstacles and humans, due to the results obtained in the simulation. The basic structure of the system is supported by the ROS software, in particular, the MoveIt! and Rviz. These tools serve both for simulations and for real applications. The obtained results allow to validate the system using the algorithms PRM and RRT, chosen for being commonly used in the field of robot path planning.

Keywords: Collaborative robots · Manipulator path planning · Collision avoidance · RGB-D

1 Introduction

The ability to predict and avoid collisions for a robotic manipulator is just one of many other perspectives of Industry 4.0. Another view of Industry 4.0 is the collaboration between robots and humans, in which humans can share the same

M. F. Silva et al. (Eds.): ROBOT 2019, AISC 1092, pp. 331–342, 2020.
https://doi.org/10.1007/978-3-030-35990-4_27

work environment without additional safety equipment. Therefore, the ability of collaborative robots to acquire the working environment and to plan (or re-plan) their own movements avoiding obstacles or humans around, is of great importance to the industrial sector.

To promote the ability to acquire the working environment, Red Green Blue Depth (RGB-D) sensors can be used [1]. These sensors help in the acquisition and perception of the environment so that the system can make the path planning with restrictions of occupied spaces. This type of device has high popularity, not only by this feature, but also by the low cost. An example of this is the Kinect sensor.

Based on the pick-and-place movements of a collaborative robotic manipulator, it is intended to develop a security system capable of avoiding the collision of manipulators with objects and humans. That is, giving anthropomorphic manipulators the ability to avoid colliding with obstacles in the work environment while performing their movements. Through the union of some tools, as shown in Fig. 1, the collaborative robot can then perform different movements to reach the target point, different from what is applied today.

Fig. 1. Tool diagram for manipulators to avoid collisions.

This paper is organized as follows. After an introduction in this section, the related work is presented in Sect. 2. Section 3 presents the system architecture whereas Sect. 4 addressees the experimental verification and system deployment. Section 5 shows the results and the paper is concluded in Sect. 6 that also presents some future work.

2 Related Work

In elaborate and delicate operations, robotic arms must be able to interact and collaborate with human beings. In this sense, [2] points out the basic resources of collaborative manipulators that perform tasks together with humans. In this way, presents the importance of the acquire of the environment of the integration of human beings with robots. However, for the definition of the robot movements, path planning should be as fast as possible [3]. Demonstrating the reduction in search time when using Genetic Algorithm (GA) for the path planning, in relation to works done by earlier heuristic calculations. Based on the master link concept, [4] demonstrates the development of software that can plan trajectories in redundant manipulators. Applying potential field algorithms to smooth path movements of manipulators.

Through voice commands and gestures, a control platform in robotic arms performs the obstacle avoidance process with a hierarchy of consistent behaviors. The collaboration process is started or finalized also by gestures and commands by sound [5]. Another system combined with the manipulator controller is developed by [6], which applies the guided vision system to determine risk situations for operators. By analyzing the triangular mesh of an object, it is possible to prevent a collision in real time by applying the kinetostatic safety field process. This method depends on the position, velocity of the body and also the shape to determine the planned spaces [7]. Through a UR5 configured with 6 Degrees Of Freedom (DOF), [8] develops a platform that compares some algorithms in situations of pick-and-place. With the platform, it is demonstrated the differences between the algorithms: Rapidly-exploring Random Tree (RRT), RRTConnect, Kinematic Planning by Interior-Exterior Cell Exploration (K-PIECE), Probabilistic Roadmaps (PRM), PRM* and Extensive Space Trees (EST). Applying the same obstacle avoidance conditions, the comparison informs the advantages and disadvantages of each algorithm as to search time. In real obstacle avoidance situations, it is demonstrated in [9] the training of a neural network system to perform movements in robotic manipulators avoiding obstacles and people in the workspace. With data from the three manipulator joints and the environment, the neural network can detect collisions within a few milliseconds.

3 System Architecture

Many applications and tools need to be combined to develop the system that acquires the environment through an RGB-D sensor. And with this union, it is possible to process the data and define the free spaces for the motion planning for robotic manipulators (virtual or real). Accordingly, Fig. 2 illustrates a simplified block diagram of the system. Each block owns the tasks that are detailed on the following topics:

- **User:** It is possible to visualize the path found, before the robotic arm executes the movements;
- **Environment:** Work environment acquired by the RGB-D sensor;
- **Command:** The user can get system data in text format, or can also perform actions manually in any process involved;
- **Rviz/MoveIt!:** With these two ROS tools it is possible to verify the performance of the system, both the movements of the robot and the acquisition of the environment. It is also possible to perform graphical movements in real and virtual situations;
- **Kinect sensor:** RGB-D sensor used to acquire the working environment;
- `move_group`: Central part of the system, this element connects all the processes involved. Its main function is to order all processes to work together;
- **Planning scene:** It receives data in the already Point Cloud format and transforms it into Octree data. With this, the system can define the free spaces to plan a path;

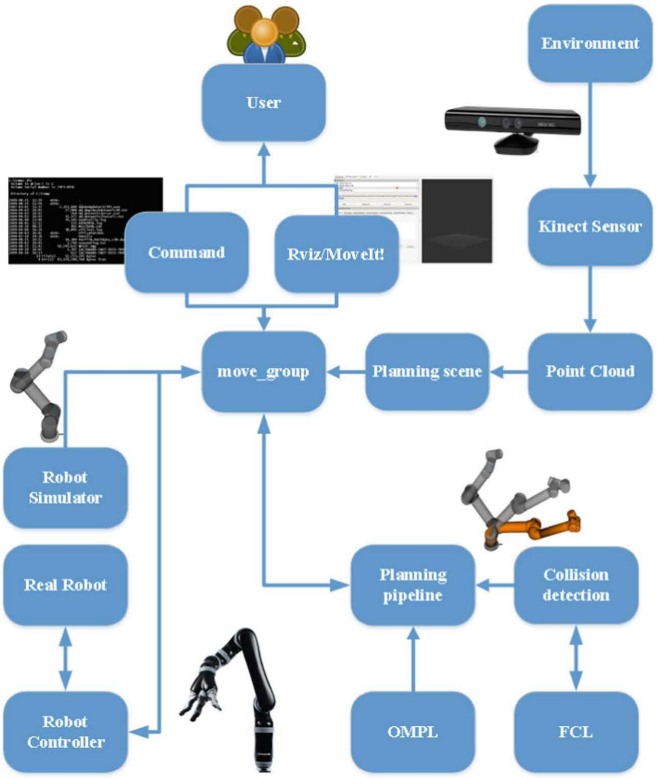


Fig. 2. Diagram of the system architecture [10].

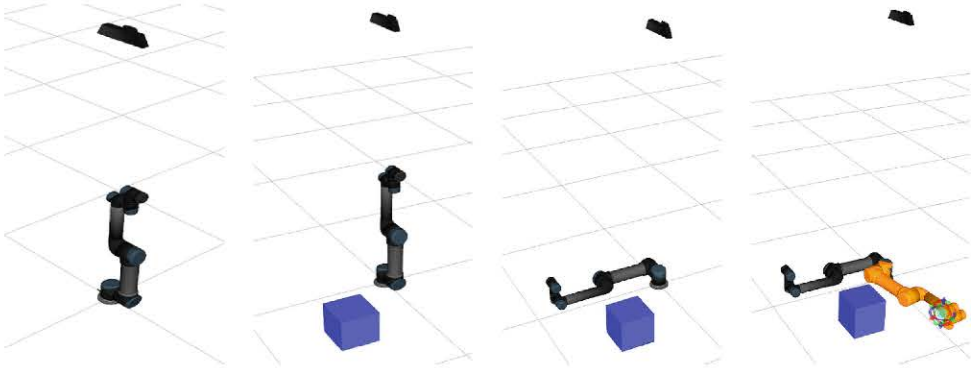
- **Point Cloud:** Process that performs the data structure of multidimensional points provided by the sensor;
- **Robot Simulator:** Trajectory simulation process, in this way it is possible to perform the movements found in the simulated robot and then apply to the real robot. It avoids possible failures or accidents;
- **Real robot:** Execution of the movements in real manipulators;
- **Planning pipeline:** Collect information from The Open Motion Planning Library (OMPL) and Flexible Collision Library (FCL), transmitting to the `move_group` the calculated trajectory and whether or not collisions occur;
- **Collision detection:** Process with the ability to verify that the manipulator is colliding with the parts of its own body and with obstacles fixed at the tip of the tool;
- **Robot Controller:** Controllers of the robot communicate with the central element of the system, in this way they can inform the current position of the robot or receive the execution of movements;
- **OMPL:** Library with algorithms that perform the calculations of trajectory in the free spaces;
- **FCL:** This library contains algorithms for calculating collisions between the manipulators and their parts, in other words, calculate whether the robot will collide or not with its own body.

As MoveIt! finds the routes that the robotic arm must make to reach the final point, it is necessary to configure an algorithm to perform the calculation. Two algorithms were chosen for this configuration, the PRM and the RRT. Both were selected because they are commonly used by the community in path planning, so these algorithms can validate the system. Still in MoveIt!, two parameters have changed: Planning time and Planning Attempt. The first parameter has been set to 10 s, this is the time the system will have to find a possible route. The second parameter is set to the value of 15, which determines the maximum number of paths that the system must find. Therefore, if the system does not find the path in these conditions, the robotic arm will not be given the command to move. All other parameters within MoveIt! are set as default.

4 Experimental Verification and System Deployment

To verify if the developed system is able to avoid collisions, a simulated environment with the UR5 manipulator and a Kinect sensor is developed. Thus, the virtual scenario can indicate if the proposed system works as expected. The virtual manipulator is expected to perform pick-and-place movements in the free spaces, that is, without colliding with obstacles in the environment. In this Section, a brief demonstration of the entire simulation check is demonstrated. For the complete demonstration see [10,11].

The simulated environment for the system verification is elaborated with a box as obstacle, UR5 and Kinect sensor. The Fig. 3 shows the sequence of steps for this analysis. Then, the UR5 is adjusted to the original position of the scene, with coordinates $(0, 0, 0)$ and the Kinect is centralized with UR5 but at 150 cm height, with coordinates $(0, 0, 150)$. The Fig. 3a shows the first positioning of the elements. For now the Kinect sensor is not acquiring data from the environment.

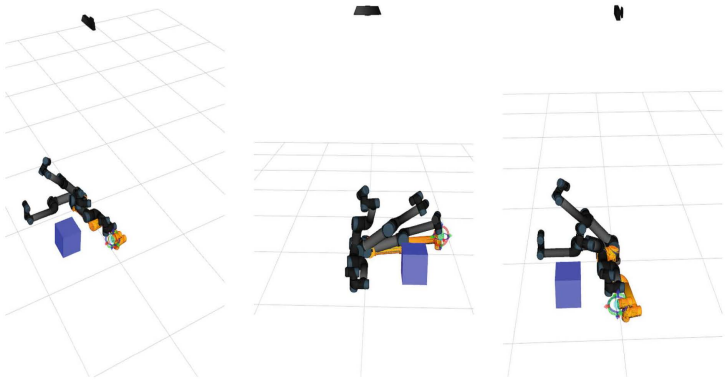


(a) First step. (b) Second step. (c) Third step. (d) Fourth step.

Fig. 3. Sequence of images that demonstrate the steps of the simulation environment to move UR5 without collisions [10].

Then a simple box is inserted into the scenario. As Fig. 3b shows, the positioning of the box is $(50, 50, 12.5)$. This is due to the dimensions of the box, as it

is configured to be 25 cm length, 25 cm width and 25 cm height. The positioning on the Z axis is half the size of the box because it is relative to its own center of mass. Then the initial and final position is defined to simulate the pick-and-place movements in UR5, as illustrated by the Fig. 3c for the initial point and Fig. 3d for the final point. The sensor is activated to acquire the data from the environment and then the proposed system is also activated to determine the path for the UR5 to reach the final point without colliding with the box. The sequence of images in the Fig. 4 shows the trajectory found.



(a) Isometric. (b) Front. (c) Side.

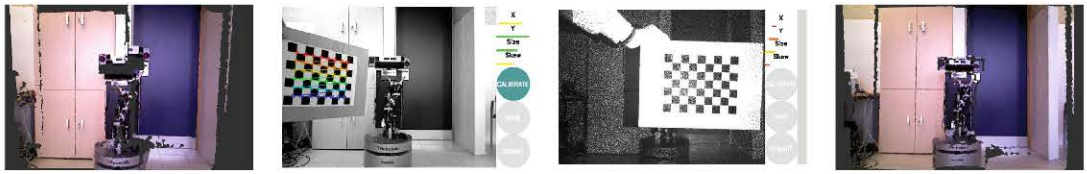
Fig. 4. Sequence of images that demonstrate from different views the simulation of the system that avoids collisions [10].

5 Results in Real Situations

After the validation of the proposed system in a simulation environment, it is possible to carry out the process of avoiding obstacles and humans in real situations, that is, to the real manipulator and his work environment. The tests in real situations of the developed system were realized with the robot manipulator Jaco2 6 DOF, with the same perspective to realize the movements of pick-and-place. Similar to the simulation steps, the created scenario is acquired through an RGB-D sensor. A foam object and a human were inserted as obstacles during the trajectory from the initial to the final point.

5.1 System Avoiding Real Obstacles with Real Manipulator

Differently from the simulation, before setting up the scenario to test the obstacle avoidance system, it is necessary to calibrate the RGB-D sensor because in simulation the sensor is used with ideal characteristics, and for the real case there is the problem of desynchronization between the RGB image and the depth image [12]. Figure 5 shows the sensor calibration steps, where it is possible to detect the desynchronization of the images in Fig. 5a.



(a) Identification of desynchronization. (b) RGB image calibration. (c) Depth image calibration. (d) Result after the calibration process.

Fig. 5. Images that demonstrate the calibration process of the Kinect sensor.

The reason for the desynchronization is due to the fact that the images are generated by different sensors, calibration methods have been extensively investigated since the first version of the Kinect sensor [12]. Depth sensors can instantly and completely represent the 3D structure of the current surrounding environment. The 3D data structure of the environment is displayed in Colored Point Cloud, and with this data, robots can perceive and interact with other agents within the workspace.

Another important factor for RGB-D sensor calibration is the synchronization of the environment viewed in Rviz and MoveIt! with real workspace, that is, the Point Cloud needs to be synchronized with real robotic arm movements. Otherwise, the proposed system may determine as obstacles the parts of the robot in the Point Cloud that are not synchronized. Therefore, sensor calibration is also a fundamental part of the ground truth of the system.

With the complete calibration of the RGB-D sensor, the working environment of the robot that is the test scenario is created. The elements for this scenario can be seen in the Fig. 6. The foam object is shown in the Fig. 6a, the sensor fixed in Fig. 6b and the Jaco2 6 DOF in Fig. 6c.

(a) Foam object. (b) Kinect fixation. (c) Front view of scenario.

Fig. 6. Images that demonstrate the real scenario.

The manipulator is attached to another mobile robot, which is only serving as the base for the Jaco2 6 DOF. Therefore, no further details will be discussed about the mobile robot, as this will not influence the test. Another factor is that the model of the manipulator is different from that applied in the simulation, however the idea is to identify if the developed system can also be applied in different models of robotic arms. Figure 7 illustrates the schematic of the scenario with the placement and dimensions of all elements inserted. A rectangular shaped foam is inserted as an obstacle into the Jaco2 6 DOF workspace, it is positioned 55 cm away from the arm, or 140 cm from the sensor.

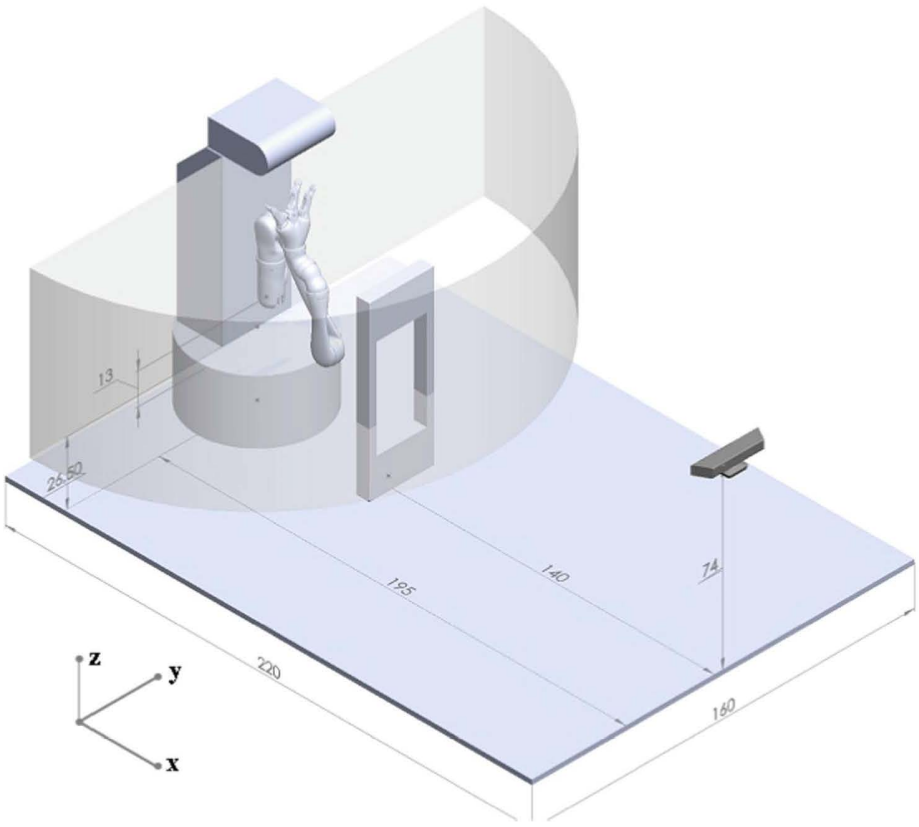
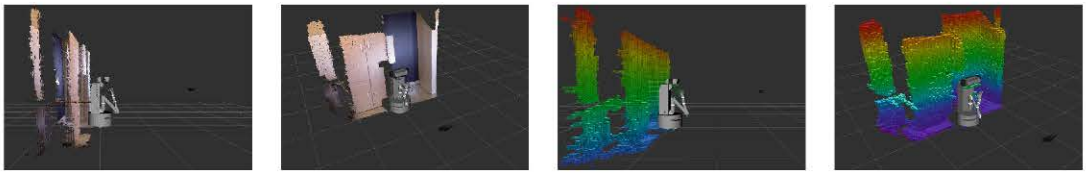


Fig. 7. Geometric arrangement of the elements with the distances, positions and workspace of the robotic arm.

After acquiring the Point Cloud, the next step is to turn these data into Octree format. In this way, the system can check the free spaces in the environment to calculate a path for the robot. The Fig. 8 shows the acquisition of the environment and the transformation to Octree at different angles. It is also possible to note that the simulation is completely synchronized with the perception from the environment.



(a) Point Cloud, side view. (b) Point Cloud, isometric view. (c) Octree data, side view. (d) Octree data, isometric view.

Fig. 8. Images that demonstrate the acquired Point Cloud and the Octree transformation.

With the scenario in Octree format, is possible to set the initial and final positions to move Jaco2 6 DOF. The Fig. 9a indicates the starting point, and in Fig. 9b the end point is indicated. These points have been chosen for the pick-and-place movements that the Jaco2 6 DOF can carry through the obstacles or avoid them. For example, with these points the path planning could collide with the obstacle if the developed system fails, in other word, this case the path planning would be perpendicular to the obstacle.

(a) Initial point. (b) Final point.

Fig. 9. Setting the initial point and the final point.

Figure 10 demonstrates this example, it is noted that with the disable sensor the arm moves in a trajectory that collides with the obstacle. There are still other possible trajectories that the robot controller can do to reach the final point, however with the aid of the sensor the motion planning can deviate from the obstacle.

(a) First state of motion. (b) Second state of motion. (c) Third state of motion.

Fig. 10. Images that show the movement with collision in the obstacle.

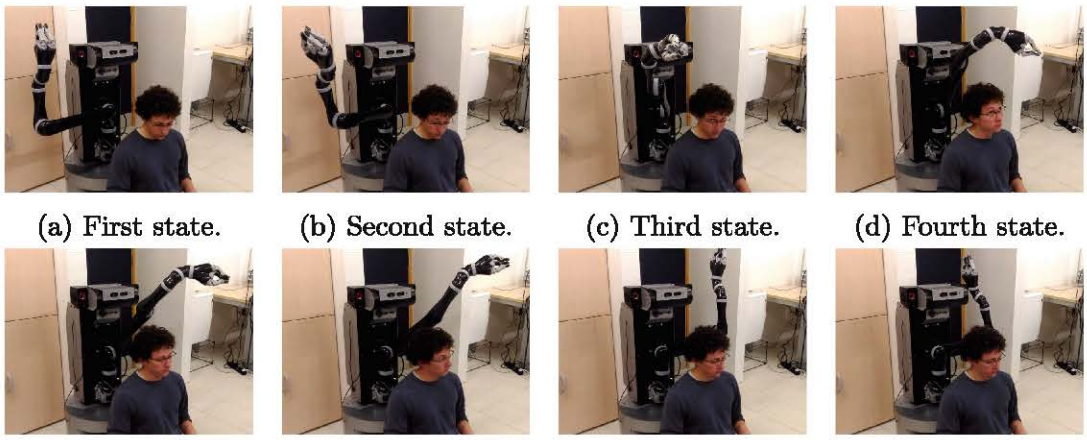
Therefore, the robotic arm does not have the ability to avoid collision without activating the RGB-D sensor. Then the developed system is activated and requested that the arm return to the initial position, since the last positioning of the robot is the end point presented in Fig. 10. In this step, the manipulator will not move if the system does not find path planning without collision. But if the system finds a trajectory in the free space, the arm will move to the desired point (initial point). The result can be viewed in the sequence of images in Fig. 11.

(a) First. (b) Second. (c) Third. (d) Fourth. (e) Fifth.

Fig. 11. Sequence of images demonstrating the movement without collision with the obstacle.

5.2 System Avoiding Human with Real Manipulator

To verify if the developed system can avoid collisions with humans, the same scenario from the previous test is used. Hence, in this test there is the replacement of the foam object by a human. In this sense, the purpose of this test is to create a real scenario with a human in the workspace of the robotic arm and perform pick-and-place movements with the Jaco2 6 DOF (the same points of the previous test) through acquisition by the sensor RGB-D. The Fig. 12 demonstrates in an image sequence the movement of the arm avoiding collision with the human. Similarly as in the previous test, the robot would not move if the system finds no trajectory or checks the collision.



(a) First state. (b) Second state. (c) Third state. (d) Fourth state.

(e) Fifth state. (f) Sixth state. (g) Seventh state. (h) Eighth state.

Fig. 12. Sequence of images demonstrating the movement without collision with the human.

6 Conclusion

In the course of this work, a system was developed capable of path planning in robotic manipulators that avoid collision with objects and humans presented in the work environment. Collaborative (real and virtual) manipulators acted as test models and two path planning algorithms used to determine the route. The system allows to re-plan the robotic movement avoiding collisions, guaranteeing the execution of the operations, with the use of ROS as the platform, RGB-D sensor (Kinect) to acquire the environment and, a foam and human as obstacle. Therefore, the system provides the proposal for robotic applications that collaborate with humans, according to the new paradigms of Industry 4.0.

Based on the results obtained in this work, future work can be done for the developed system. In such sense, optimizing the Point Cloud acquisition can lead to less delay in robot movements, an implementation of a second RGB-D sensor to achieve a more uniform working environment and finally, the determination of an optimal resolution for the generation of Octree based on computational costs are pointed out.

Acknowledgments. This work has been partially funded by Junta de Castilla y León and FEDER funds, under Research Grant No. LE028P17 and by "Ministerio de Ciencia, Innovación y Universidades" of the Kingdom of Spain through grant RTI2018-100683-B-I00.

References

1. Mutto, C.D., Zanuttigh, P., Cortelazzo, G.M.: Time-of-Flight Cameras and Microsoft Kinect (TM). Springer, Boston (2012). ISBN: 1461438063978146143806 9
2. Khatib, O., Yokoi, K., Brock, O., Chang, K., Casal, A.: Robots in human environments: basic autonomous capabilities. Int. J. Robot. Res. **18**(7), 684–696 (1999)
3. Ali, M.A.D., Babu, N.R., Varghese, K.: Collision free path planning of cooperative crane manipulators using genetic algorithm. J. Comput. Civ. Eng. **19**(2), 182–193 (2005)
4. Conkur, E.S.: Path planning using potential fields for highly redundant manipulators. Robot. Auton. Syst. **52**(2–3), 209–228 (2005)
5. De Luca, A., Flacco, F.: Integrated control for pHRI: collision avoidance, detection, reaction and collaboration. In: 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), pp. 288–295. IEEE (2012)
6. Wang, L., Schmidt, B., Nee, A.Y.: Vision-guided active collision avoidance for human-robot collaborations. Manuf. Lett. **1**(1), 5–8 (2013)
7. Polverini, M.P., Zanchettin, A.M., Rocco, P.: A computationally efficient safety assessment for collaborative robotics applications. Robot. Comput. Integr. Manuf. **46**, 25–37 (2017)
8. Frutuoso, I.P.: Smart collision avoidance system for a dual-arm manipulator. Master thesis (2018). https://hdl.handle.net/10216/114146
9. Sharkawy, A.N., Koustoumpardis, P.N., Aspragathos, N.: Human-robot collisions detection for safe human-robot interaction using one multi-input-output neural network. Soft Comput. 1–33 (2019)
10. Brito, T.: Intelligent collision avoidance system for industrial manipulators. Master thesis (2019). http://hdl.handle.net/10198/19319
11. Brito, T., Lima, J., Costa, P., Piardi, L.: Dynamic collision avoidance system for a manipulator based on RGB-D data. In: Iberian Robotics conference. pp. 643–654. Springer, Cham (2017)
12. Darwish, W., Tang, S., Li, W., Chen, W.: A new calibration method for commercial RGB-D sensors. Sensors **17**, 1204 (2017). https://doi.org/10.3390/s17061204

Source details

Advances in Intelligent Systems and Computing

Formerly known as: Advances in Intelligent and Soft Computing

Scopus coverage years: from 2005 to 2006, 2008, 2010, from 2012 to Present

Publisher: Springer Nature

ISSN: 2194-5357

Subject area: Computer Science: General Computer Science | Engineering: Control and Systems Engineering

View all documents > | Set document alert | Save to source list

CiteScore 2018
0.54
Add CiteScore to your site

SJR 2018
0.174

SNIP 2018
0.434

CiteScore | CiteScore rank & trend | CiteScore presets | Scopus content coverage

CiteScore 2018

Calculated using data from 30 April, 2019

$$0.54 = \frac{\text{Citation Count 2018}}{\text{Documents 2015 - 2017*}} = \frac{\text{7162 Citations}}{\text{13 364 Documents}}$$

*CiteScore includes all available document types

View CiteScore methodology > CiteScore FAQ >

CiteScore rank

Category	Rank	Percentile
Computer Science — General Computer Science	#156/206	24th
Engineering — Control and Systems Engineering	#191/233	18th

View CiteScore trends >

CiteScoreTracker 2019

Last updated on *06 February, 2020*
Updated monthly

$$0.54 = \frac{\text{Citation Count 2019}}{\text{Documents 2016 - 2018}} = \frac{\text{10 162 Citations to date}}{\text{18 717 Documents to date}}$$

Metrics displaying this icon are compiled according to Snowball Metrics ↗ , a collaboration between industry and academia.