electronics

MDPI

Article

Intrinsically Distributed Probabilistic Algorithm for Human–Robot Distance Computation in Collision Avoidance Strategies

Marcello Chiurazzi [1,2,*], Alessandro Diodato [1], Irene Vetrò [1], Joan Ortega Alcaide [1,2], Arianna Menciassi [1,2] and Gastone Ciuti [1,2]

[1] The BioRobotics Institute, Scuola Superiore Sant'Anna, 56127 Pisa, Italy; alessandro.diodato@santannapisa.it (A.D.); irevetro@gmail.com (I.V.); j.ortegaalcaide@santannapisa.it (J.O.A.); arianna.menciassi@santannapisa.it (A.M.); gastone.ciuti@santannapisa.it (G.C.)

[2] Department of Excellence in Robotics & AI, Scuola Superiore Sant'Anna, 56127 Pisa, Italy

* Correspondence: marcello.chiurazzi@santannapisa.it; Tel.: +39-335-6979421

Received: 20 February 2020; Accepted: 21 March 2020; Published: 25 March 2020

Abstract: Humans and robots are becoming co-workers, both in industrial and medical applications. This new paradigm raises problems related to human safety. To accomplish and solve this issue, many researchers have developed collision avoidance strategies—mainly relying on potential field approaches—in which attractive and repulsive virtual forces are generated between manipulators and objects within a collaborative workspace. The magnitude of such virtual forces strongly depends on the relative distance between the manipulators and the approaching agents, as well on their relative velocity. In this paper, authors developed an intrinsically distributed probabilistic algorithm to compute distances between the manipulator surfaces and humans, allowing tuning the computational time versus estimation accuracy, based on the application requirements. At each iteration, the algorithm computes the human–robot distances, considering all the Cartesian points within a specific geometrical domain, built around humans' kinematic chain, and selecting a random subset of points outside of it. Experimental validation was performed in a dynamic and unstructured condition to assess the performance of the algorithm, simulating up to six humans into the shared workspace. Tests showed that the algorithm, with the selected hardware, is able to estimate the distance between the human and the manipulator with a RMSE of 5.93 mm (maximum error of 34.86 mm).

Keywords: collaborative robotics; perception; human–robot collaboration; probabilistic algorithms; collision avoidance strategies

1. Introduction

Humans and robots are becoming co-workers in a wide variety of fields, either in industrial [1] or medical [2] scenarios. Robotic platforms have been developed and are already employed to improve repeatability and accuracy in computer-assisted surgery; a significant example is represented by high-intensity focused ultrasound therapy, where human and robots constantly share the same operating environment [2]. During recent decades, many researchers have focused their efforts on understanding and characterizing human–robot interaction and collaboration. In this context, several collaborative strategies, both hardware and software, have been developed, aiming at guaranteeing the human safety in a robotic collaborative environment. Keeping humans safe is necessary, especially in case of dynamic and unstructured environment. To this purpose, in fact the ISO/TS 15066: 2016 has been published and used, as a technical specification in order to define guidelines regarding the maximum transferrable energy arising from a collision between a robotic machine and a human being.

These strategies can be classified into two main categories, i.e., pre- and post-contact strategies [3]. Post-contact strategies cannot avoid collision, but they can avoid injuries by limiting the energy transmitted from robots to humans (as specified in the ISO/TS 15066:2016). For this purpose, post-contact strategies are usually based on the design of machines capable of accumulating and/or dissipating the energy arising from the collision [4,5]. The main drawback of this approach is that collision can be muffled but not avoided. Moreover, if the collision cannot be prevented, the employed robotic arm must have a light weight and move at low speeds. On the other hand, pre-contact strategies are employed to prevent a collision before it occurs. These strategies are mainly based on potential field approaches, which use distances and relative velocities between obstacles and manipulators to create virtual repulsive and attractive forces [6]. On this purpose, calculating these distances in an accurate, robust and time-efficient manner is crucial for a safe human–robot collaboration (HRC). Nowadays, these distances are mostly computed through vision systems that have widely demonstrated their efficiency in object/human detection and reconstruction [7].

1.1. Prior Work

To estimate the human position in a factory cell, Ragaglia et al. [8] propose a stochastic methodology, merging external condition and human gait model with human recognition and tracking algorithms. The robotic factory cell includes several cameras and it is divided into co-existence, co-operation and interference areas. The developed algorithm uses a predictive model based on an offline unsupervised human path classification to estimate the area the human operator will move to, associating a likelihood for each of the areas. The human being is modeled with a rectangular box. On the other hand, Morato et al. [9] uses a skeleton-like model to track the human and estimating the position of the skeleton joints using a Kalman filter approach that merges the information of multiple Microsoft Kinects cameras (Microsoft Corp., Redmond, Washington, USA). Humans are modeled using virtual spheres centered in the midpoint between two consecutive human joints: the diameter of the sphere is equal to the distance between the two joints. Modeling human bodies with spheres has also been proposed by Kulic et al. in [10], where the information is merged with physiological sensors to estimate the user's affective state. Safeea et al. proposed to reconstruct humans and manipulators using cylindrical primitives, presenting a closed-form solution to compute distances between them [11]. In this study, a case study where a human is represented by one cylindrical primitive and the manipulator is composed of two cylindrical primitives, was presented. Using more cylindrical primitives increases the fidelity of human and manipulator reconstruction, as much as the model complexity leading to a computationally expansive representation.

All the techniques mentioned so far aim at reconstructing humans by exploiting convex approximations of their shape. Phan and Ferrie [12], instead, propose a model based on geodesic distance graph model to compute a high-resolution reconstruction of humans. However, the algorithm requires a computational time of 0.6 s, that is not always suitable for applications demanding shorter reaction time, such as in HRC. To reduce the computational time, Cefalo et al. [13] propose a real-time collision algorithm based on the parallel computing of Graphics Processing Units (GPU). The algorithm subtracts the depth image acquired from a camera to a virtual depth image built using CAD models. A collision is detected whenever there is a difference between the two images, i.e., whenever the robot touches or is covered by an obstacle. In Cefalo et al. [13] a single camera is used; thus, results may be affected by shadows problems. Whenever an object is observed by a camera, a blind volume, where the camera is essentially senseless, is generated. Such a phenomenon is known as the shadow effect. For this reason, the entire shadow cone generated by an obstacle is often considered as part of the obstacle itself. This assumption leads to a substantial reduction of the manipulator workspace, affecting its performances. Multiple depth cameras can be employed to overcome this problem, adding multiple points of view and, therefore, mitigating the problem of shadows. Although, a multi camera approach can solve the issue of shadow effects, it will require a high computational cost due to the fusion of the information coming from different sensors. Fisher et al. in [14] propose a master-slave architecture

of distributed cameras. The architecture has five processing layers that corresponds to the data type transferred between slave and master cameras. These processing layers vary from 0 to 5 (from raw image acquisition up to minimum distance calculation). The proposed approach, first, remaps the depth sensors information into the Cartesian space, and then clusters the information and computes the object convex hull, which is used to compute the distance. The quality of approximation of the detected objects depends on distributed processing level: the lower the level, the higher the quality will be. The average computational time of the algorithm with the lowest quality of approximation (i.e., level 5) is about 80 ms (i.e., 12.5 Hz). Moreover, the computational time increases proportionally with the number of depth sensors when distributed processing level is between 0 and 4, hampering the use of additional sensor units. In this regard, Flacco et al. faced the integration of multiple depth sensors in [15] extending his previous work [16], where the distances between control points on the manipulator surfaces and moving obstacles (including humans) were evaluated based on the concept of depth space. The presented approach assumes that a main camera covers the entire collaborative workspace. Secondary cameras, monitoring sub-regions of the workspace, are queried only when the main camera information is not enough. However, it is worth mentioning that this assumption may not be feasible in a real environment.

For the sake of clarity, Table 1 summarizes the main features of the methodologies currently used for HRC, relying on external vision cameras.

Table 1. Comparative analysis of the SoA methodologies for human–robot collaboration (HRC) using external vision cameras.

References	Hardware	Shape for Human Reconstruction	Computational Time (ms)	Mean Distance Error (mm)*
Ragaglia et al. [8]	Multiple cameras	Rectangular	880–2720	N/A
Morato et al. [9]	Multiple cameras	Spherical	N/A	10
Kulic et al. [10]	None	Spherical	N/A	N/A
Safeea et al. [11]	Single camera	Cylindrical	N/A	N/A
Phan et al. [12]	Multiple cameras	Spherical (geodesic distance)	600	149
Cefalo et al. [13]	Single camera	None	20	300
Fisher et al. [14]	Multiple cameras	None	80 (increasing with number of layers)	N/A
Flacco et al. [16]	Multiple cameras	Spherical	3.3	40

* if not available as a number in the manuscript, it was derived by analysing images and graphs.

1.2. Aim and Organization of the Work

The integration of multiple depth sensors to compute the distance between humans and robots is a crucial issue in the field of HRC. Indeed, having multiple points of view increases the robustness of human detection (in case of sensor fault) and mitigates the problem of shadows. Many developed methods in the literature first elaborate the depth image to reconstruct and/or to approximate the human shapes and, subsequently, calculate the distance between the computed shapes and the manipulators, usually with a significant computational cost.

Our idea, instead, still relying on a visual approach, is aimed at reducing the overall computational time by avoiding the reconstruction of the human shapes and directly searching the minimum distances on the depth information. This solution led to developing a method able to compute distances between manipulator surfaces and humans that is intrinsically distributed and decentralized and relies on the usage of a closed-loop algorithm for distance probability estimation. Therefore, this approach can be extended to N generic depth sensors, each of them provided with a dedicated host PC for running an independent instance of the proposed algorithm. It can calculate distances (or at least a bounded estimation of them) between humans and manipulator shapes at a specific frequency (i.e., 30 Hz with the hardware used in this study), allowing tuning the overall computational time versus the estimation

precision (i.e., computed distance error) based on the application requirements. It is important to stress that the computational cost of the proposed solution only relies on the sensor (i.e., depth camera) used to extrapolate the depth information. The entire system is decentralized; thus, multiple cameras can be used for increasing the resolution without affecting the overall computational costs.

The paper is organized as follows: Section 2 describes the software architecture and the implementation of a distance probability estimation (DPE) algorithm; then, an analytical conservative (worst-case) analysis of the DPE algorithm performance was conducted and experimental tests were performed to evaluate the intrinsically distributed probabilistic algorithm for human–robot distance computation. The obtained results are shown and discussed in Sections 3 and 4, respectively. Finally, conclusions and future work are reported in Section 5.

2. Materials and Methods

2.1. System Flowchart

A comprehensive software architecture of the HRC algorithm is illustrated in Figure 1. An integrated environmental scanner provides the depth image to the human detection algorithm, which computes the depth measurements of the humans in the workspace (Section 2.2). Then, the computed depth measurements of each human represent the input for the DPE algorithm, which uses a custom-made collision and proximity simulator (CPS) to compute the distances between depth points and manipulator surfaces (Section 2.3). Finally, these distance information is processed and stored by the DPE (Section 2.4) that, once concluded, remains in an idle state until additional depth data are available. Evaluation tests are finally described in Section 2.5.

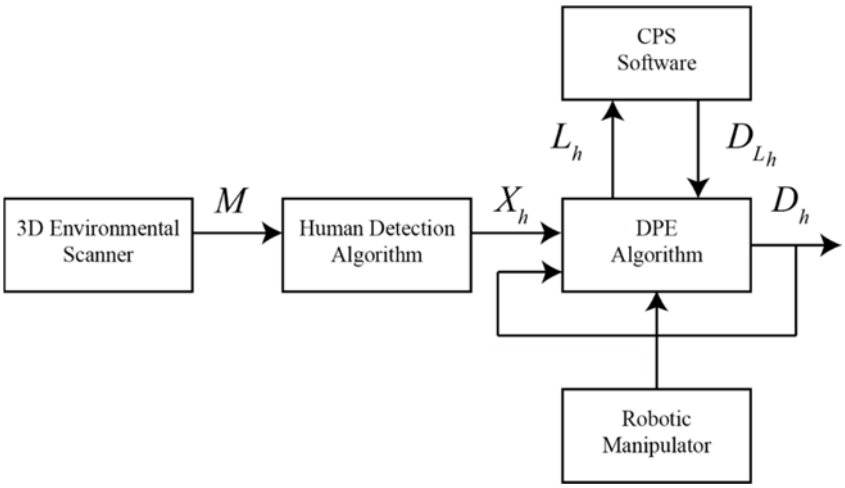

Figure 1. Software architecture diagram and data flow schematics.

2.2. Environmental Scanner and Detection of Humans

The environmental scanner used in the presented work is the Microsoft Kinect v2.0 by Microsoft Corporation (Redmond, Washington, USA) developed to allow enhanced physical interactions in gaming. The low-cost and available Software Development Kit (SDK) has made it immediately popular for many other application fields, such as robotics and computer vision. The Kinect sensor is able to simultaneously track up to six humans exploiting depth image measurements. A depth image is a matrix of distance values between the origin of the depth sensors and the closest objects/humans along defined directions. Each value in the matrix is a depth point and it represents a physical Cartesian point. Further details about the concept of depth space can be found in [16]. The accuracy and repeatability of depth measurements performed by Microsoft Kinect v2.0 are assessed in [17] and [18], which concluded that the measurement error distribution has a standard deviation within 6 mm, over

time. The developed proposed software reads depth images at 30 Hz (max frame rate) and classifies the tracked humans in subsets of Cartesian points (exploiting the C++ Kinect SDK functions). These subsets of Cartesian point (i.e., humans) will then be processed by the DPE algorithm. Although, the Microsoft Kinect sensor has been selected and used in this study as a gold standard technology, any depth image sensors can be used with the presented intrinsically distributed probabilistic algorithm. Figure 2 reports a schematic representation of the collaborative multi-agent environment set-up implemented and used for testing the DPE algorithm. The relative starting position arises from the assumption that the robot has the operative working space in the frontal sector and that the human approaches such workspace from the opposite side. The proposed method assumes that the pose obtained by the depth sensor (LRF_s) is intrinsically and extrinsically calibrated with respect to the robotic arm (GRF) so that the error, introduced by such calibration, can be assumed to be neglectable with respect to the error introduced by the depth information provided by the sensor itself. On the other hand, the resolution provided by the proposed methodology intrinsically depends on the resolution provided by the depth sensor in positioning the subset of Cartesian points representing the humans. In practical terms, the method proposed is independent by the sensor used as it assumes a neglectable positioning error of such subset of points. If the sensor employed for computing the depth information introduces an error, which will depend on the sensor resolution at the distance where the human is positioned, such error is in the worst case (when the error is in the minimum distance direction) directly added to the minimum distance $\left(d_h^m\right)$ computation error.

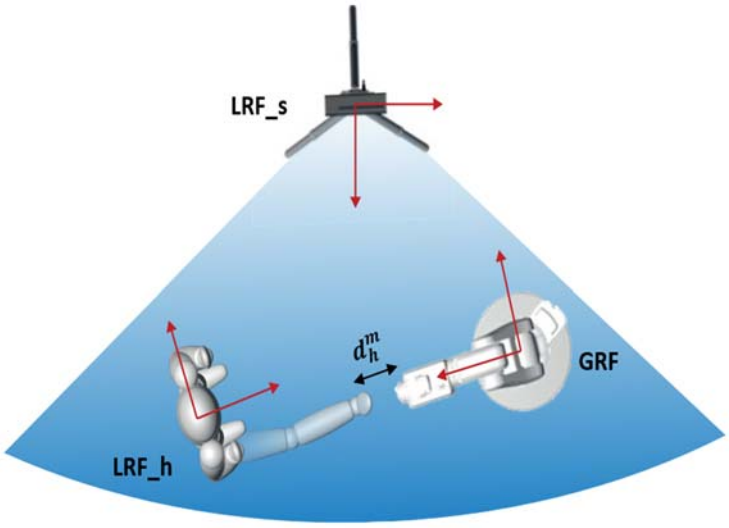

Figure 2. Multi-agent environment made up by: anthropomorphic robot (Global Reference Frame), human being (Local Reference Frame_human), and depth sensor (Local Reference Frame_sensor; d_h^m is the human–robot minimum computed distances.

2.3. Collision and Proximity Simulator

In order to compute the distances between the selected Cartesian points in L_h and the manipulator surfaces, a CPS was developed by the authors; it performs a virtual reconstruction of both the manipulator and the working environment, creating, moving and deleting virtual meshes (i.e., spheres). Although many robotic simulators have already been developed (e.g., Gazebo -http://gazebosim.org), the authors decided to develop a new simulator to entirely customize the algorithm for tuning its performances in terms of precision and computational time. The CPS was developed in C++ programming language. The simulator is based on the Bullet Physics engine (www.pybullet.org/wordpress), and it has been developed following the performance analysis presented by Boeing et al. in [19]. Bullet Physics engine is multi-platform; thus, it is possible to compile the code under Linux, Windows, and Macintosh operating systems. However, the Bullet Physics is only capable of simulating

rigid objects and does not allow to visualize the entire virtual scenario. An open source viewer (Ogre3D - www.ogre3d.org) was integrated to implement the visualization of the simulated scenario. The execution of the viewer rises the computational cost of the overall simulator; for that reason, visualization can be online activated or deactivated. The CPS is implemented following a client-server paradigm; the server is represented by the simulator that accepts a connection by a client application. Client applications can be written in arbitrary programming languages, and they are interfaced with the CPS through a simple dedicated interface (e.g., socket applications). The command interface allows to create, move and delete meshes and to check the distances between meshes. The command interface allows multiple clients to request the simulator to either adapt the 3D virtual scenario (using exteroceptive sensors information) or to check the distances between items. It is worth noting that the developed simulator is modular and multi-platform. Modularity enables its use in distributed control architectures (i.e., control algorithms run in different machines) distributing and parallelizing the computational cost of the algorithm, whereas multi-platform compatibility allows implementing applications in most of the operating systems.

The Bullet engine calculates the distance between two objects using the Gilbert–Johnson–Keerthi algorithm, which finds the minimum distance between two convex sets [20]; thus, manipulators and depth points must be modeled with convex meshes. Manipulators are modeled as an assembly of rigid links connected by rigid transformation matrices based on the Denavit–Hartenberg convention [21]. Usually, the models of the manipulator links, provided by the manufacturer, are excessively detailed for proximity computations. Therefore, in this work, the different links models have been simplified in order to minimize the number of triangular elements in the mesh by deleting: (I) concave areas, (II) constructive details, (III) logos and commercial details, and (IV) by approximating the volumes of the objects with simple convex polygons; simplifying the manipulator links models is a common practice in the state of the art [22]. The outcome of the simplified model is represented in Figure 3, where approximately 200 triangular elements are used to model the manipulator. Although these approximations might entail a loss of the overall safety, the simplified models envelop the real model and, therefore, the computed distances are always smaller or equal than the actual correct distance. The mesh type used to represent the Cartesian points is spherical, which has a faster and more efficient algorithm for computing the distance with respect to a triangular element. The CPS accepts as input the manipulator joint angles for updating the manipulator pose, and it computes all the distances (with point of application and orientation) between all manipulators links and all the created spheres (i.e., Cartesian points). The output is a list of vectors that represents the computed distances between the spheres and each robotic link.

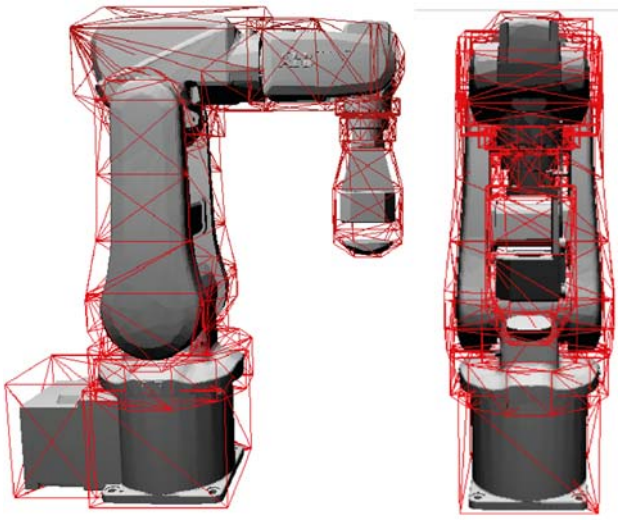

Figure 3. ABB IRB 120 manufacturer model together with the triangular-type element meshes used to compute the minimum distance to the manipulator in the collision and proximity simulator.

2.4. Distance Probability Estimation Algorithm

The Microsoft Kinect framework provides a subset X_h of Cartesian points representing the human. For each human representation h, the algorithm creates and fills a list L_h with N_h Cartesian points of the subset X_h. The first algorithm iteration fills the L_h list with N_h random points from the subset X_h. On the subsequent iterations, the algorithm starts filling the list L_h with all the points of X_h that are inside a square (i.e., proximity square) of (2W + 1) image pixels side centered at the last computed minimum distance D_h. We define N_{s_h} as the number of X_h points found inside the square. Then, the algorithm randomly picks $N_{r_h} = N_h - N_{s_h}$ points outside the proximity square. Once the list L_h is full, all the distances $D_{L_{hj}}$ between the manipulator surfaces and the points in L_h are computed by means of the CPS (Section 2.4). D_h is defined as the minimum value in the list D_{L_h}. Once all subsets X_h are processed, the DPE algorithm waits for the next depth image M acquisition. Figure 4 presents the pseudo-code of the DPE algorithm. The algorithm was developed in C++ programming language, making use of standard C++ libraries.

Data: M depth image, D_h previous minimum distance with relative indexes i_{Mh} and j_{Mh}.
index=0
for $h = 1$ to *Number of Human* **do**
empty the list L_h;
for i = -W **to** W **do**
for j = -W **to** W **do**
if $M(i + i_{Mh}; j + j_{Mh}) \in X_h$ **then**
insert $M(i + i_{Mh}; j + j_{Mh})$ into L_h;
index=index+1
end
end
end
while *index* < N_h **do**
app = randomly chose an element in X_h / L_h;
insert app to L_h;
index = index+1;
end
D_{L_h} = evaluate distances on L_h;
$[D_h, i_{Mh}, j_{Mh}]$ = find minimum on D_{L_h};
end

Figure 4. Pseudo-code of the Distance Probability Estimation (DPE) algorithm.

The worst-case scenario for the algorithm occurs when the minimum is in a random position outside the proximity square. This situation happens at the beginning of the DPE algorithm execution or when the algorithm estimates a wrong distance in the previous step, i.e., the location of the estimated distance in X_h is different from the actual location of the minimum distance. Therefore, only the randomly picked N_{r_h} Cartesian points can estimate correctly the distance between the human and the manipulator. The error e_h is introduced in (1) to provide an analytical analysis of the DPE performance in the conservative worst-case scenario. Terms d_h^M and d_h^m are the maximum and the minimum distances

of the Cartesian point subset X_h with respect to the manipulator surfaces, while D_h is the distance estimated by the DPE algorithm.

$$e_h = \frac{D_h - d_h^m}{d_h^M - d_h^m}. \tag{1}$$

The error e_h is a random variable with a finite support (the interval support is [0,1]). It is worth investigating the cumulative distribution function P ($e_h \leq \varepsilon$), which expresses the probability that the estimated distance D_h is lower than $d_h^m + \varepsilon\left(d_h^M - d_h^m\right)$.

Without loss of generality, the human–manipulator distances are normalized to the interval [0,1], being $d_h^m = 0$ and $d_h^M = 1$. It is then clear that e_h follows the same distribution of D_h. Since the D_h distribution is not known a priori, D_h is defined as the minimum value out of N_{r_h} values uniformly randomly distributed in [0,1]. The probability distribution of D_h responds to a Beta distribution with parameters a = 1 and b = N_{r_h}. Equation (2) and (3) expresses P ($e_h \leq \varepsilon$) as function of the number of points N_{r_h} picked outside the proximity square. Figure 5 illustrates the cumulative distribution functions P ($e_h \leq \varepsilon$) for some potential values N_{rh} to be used.

$$\mathrm{P}(e_h \leq \varepsilon) = I_x\left(1, N_{r_h}\right) = \frac{B\left(x; 1, N_{r_h}\right)}{B\left(1; 1, N_{r_h}\right)}. \tag{2}$$

where.

$$B(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1}dt. \tag{3}$$

Equation (2) shows that P ($e_h \leq \varepsilon$) does not depend on the cardinality of the subset X_h and, therefore, the performance of the DPE algorithm does not depend on the resolution of the environmental scanner or on the distance between the scanner and humans.

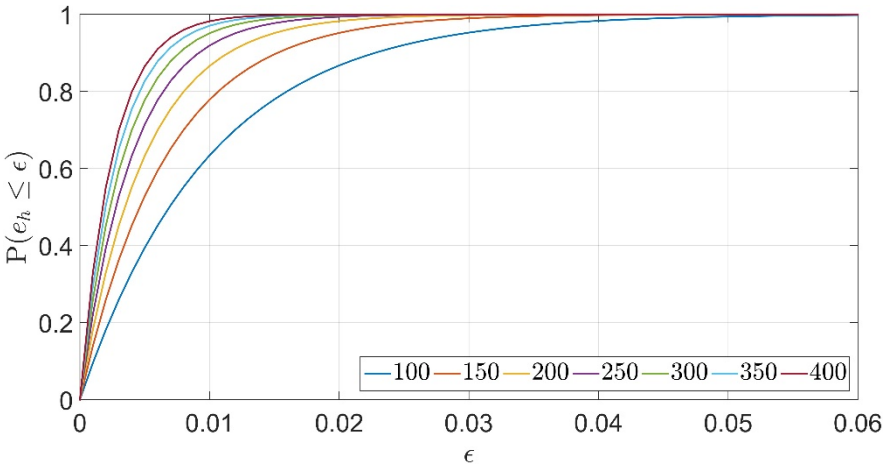

Figure 5. Cumulative distribution function of e_h when no information is not known a priori—worst case scenario—for $N_{r_h} \in \{100,150,200,250,300,350,400\}$.

2.5. Experimental Validation

In order to assess the computational cost of the CPS, a client application queries the simulator for creating, moving and deleting obstacles (i.e., 2 mm radius spheres) and checking their proximity to the manipulator links. This evaluation test will define the maximum number of depth points per human that can be evaluated, respecting the computational constraint. The simulation is structured as follows: (I) creation of n_0 obstacles, (II) randomly set the obstacles positions, (III) check the proximity, and (IV) delete the created obstacles. This simulation was conservatively performed 1000 times for each of the n_0 obstacles embedded in the simulation framework. The n_0 obstacles vary from 50 to 3000 with a

fixed step of 50. The computational times were recorded (from the client side) and the average of these times is used as the measurement reference.

The DPE algorithm may find the next minimum distance point either in the proximity square or outside of it. Two real-life experiments have been designed to evaluate the algorithm. A first one where the minimum distance point is expected to be found most of the times in the proximity square (one-handed test) and a second one forcing a spatial discontinuity of the minimum distance point position (two-handed test). For each test, 30 s of acquisition (equivalent to 900 acquired depth frames) were recorded. Tests were performed with one human positioned in front of the depth camera system, entirely inside the matrix M. The first test (one-handed test) starts with one human hand retracted at shoulder height; then the hand is moved forward, remaining at its elongated position for approximately 2 s, and is then moved back to the initial position for approximately 2 more seconds. The cycle was repeated six times. The second test uses both human hands. It starts with one arm outstretched forward and the other one retracted and aligned with the chest. Both hands are at the shoulder height. Then, both arms are moved back and forth continuously and in counter phase, always at the shoulder height. In this way, once both hands are at the same distance with respect to the manipulator, a discontinuity in the minimum distance point location is generated in the next time step.

The performance of the DPE algorithm (i.e., the error between the distance estimated by the DPE algorithm and the minimum distance) was assessed in a conservative worst-case scenario, which occurs when the maximum number of detectable humans (i.e., 6 subjects) are within the workspace. One hundred instances of DPE algorithm were performed for each experimental dataset (i.e., one- and two-handed depth data). For each DPE execution i, we estimated the vector D_h(t) that is the collection of the distances D_h estimated by the DPE algorithm for each depth image. Therefore, the vector D_h(t) is composed of 900 values and it is a function of time. Following the worst-case approach, we considered the maximum distance between human and manipulator estimated by the DPE algorithm at each moment among all the DPE execution; this vector is defined by Equation (4).

$$D_m(t) = \max_i(D_h^i(t)). \tag{4}$$

All the simulations were performed with a PC (under Windows operating system) with an Intel Core i7-6700 processor and 16 GB of RAM. The client application and the collision and proximity simulator run on the same machine.

3. Results

Figure 6 reports the average of the computational time required to create and delete obstacles as a function of the number of spheres and the computational constraint of our application (i.e., 33.3 ms, maximum sampling period of the Microsoft Kinect camera v2.0). It was observed that the computation time for deleting the spheres grows with the square of its number, violating the computational time constraint at 1500 spheres. The computational times for moving the spheres (yellow curve) and for computing the proximity (red curve) are depicted in Figure 7. In this case, the computational time grows linearly with the number of spheres. The results of these tests suggest creating all the required spheres when the client application starts, instead of creating and deleting the spheres at each iteration; all the spheres will be deleted when the client application shuts-down. At each iteration, the client application acquires and processes the depth information, moves the spheres and finally computes their distances with respect to the manipulator surfaces. The time to acquire and process the depth information was estimated to be about 10 ms. Since the computational constraint is 33.33 ms, the remaining time to move the spheres and compute the distances is equal to 23.33 ms. Given this time constraint, the simulator can move and compute the distances of approximately 2400 spheres (Figure 7). To reconstruct humans, all the spheres will be uniformly distributed among humans in the workspace (i.e., 2400 divided by the number of humans). Therefore, each human will be covered with N_h = 400 spheres when the maximum number of detectable humans (i.e., six humans) is within the workspace.

As mentioned before, we choose to study the performance of the DPE algorithm in a conservative worst-case scenario. As a specification, we will require P(e_h > 0.05) to be smaller than 1e−5 (equivalent to expect an error greater than 5% after more than 1 h of depth frame acquisition). Following (2), this specification leads to set the parameter W equal to 6.

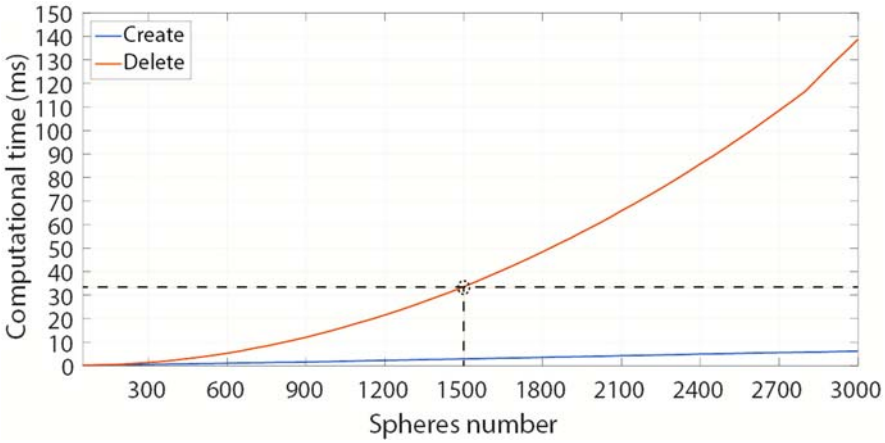

Figure 6. Computational time required to create and delete spheres with the CPS. The computational time constraint (i.e., 33.3 ms) of the application is also depicted with a dashed line.

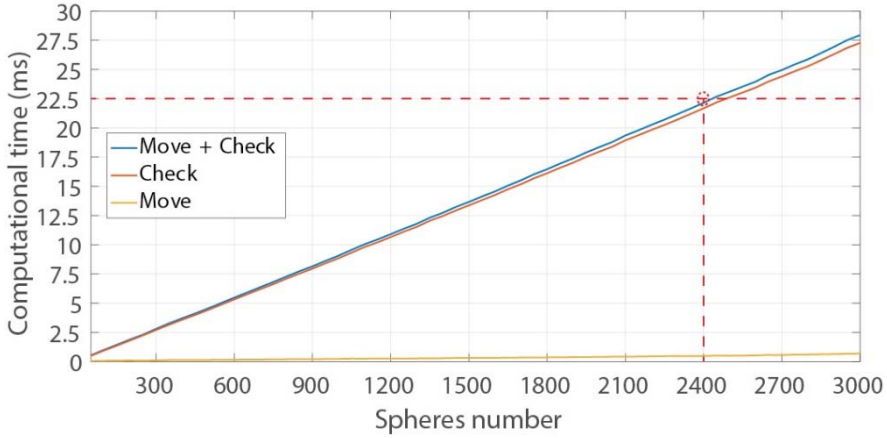

Figure 7. Computational time required to move the spheres and calculate the sphere distance to the manipulator. The dashed line corresponds to the imposed time constraint.

The first test employs only one hand. The results are shown in Figure 8 where the maximum estimated distance D_M(t) is plotted vs. the minimum distance D_h^m(t). Clearly D_M(t) always overestimates D_h^m(t). The Root Mean Square (RMS) error between the two vectors is 2.78 mm with a maximum value of 26.29 mm. The maximum distance estimation error leads to a maximum relative error e_h of 3.52%. In addition, the DPE algorithm estimates the exact distance 79.78% of the times (difference between simulated and computed distance is 0, i.e., exact distance, considering the intrinsic accuracy error of the depth sensor [17]). The results of the two-handed test are reported in Figure 9. In this case, the maximum value of the error between D_M(t) and D_h^m(t) is 34.86 mm (e_h equal to 3.89%) with an RMS of 5.93 mm. In this case, the algorithm estimates the exact distance 53.44% of the times. Figure 10 presents a zoom-in at the points, where the DPE algorithm presents the maximum estimation error. Namely, Figure 10a reports the one-handed test, whereas Figure 10b details the two-handed test. Both graphs highlight that the DPE algorithms recover the estimation error in less than 300 ms.

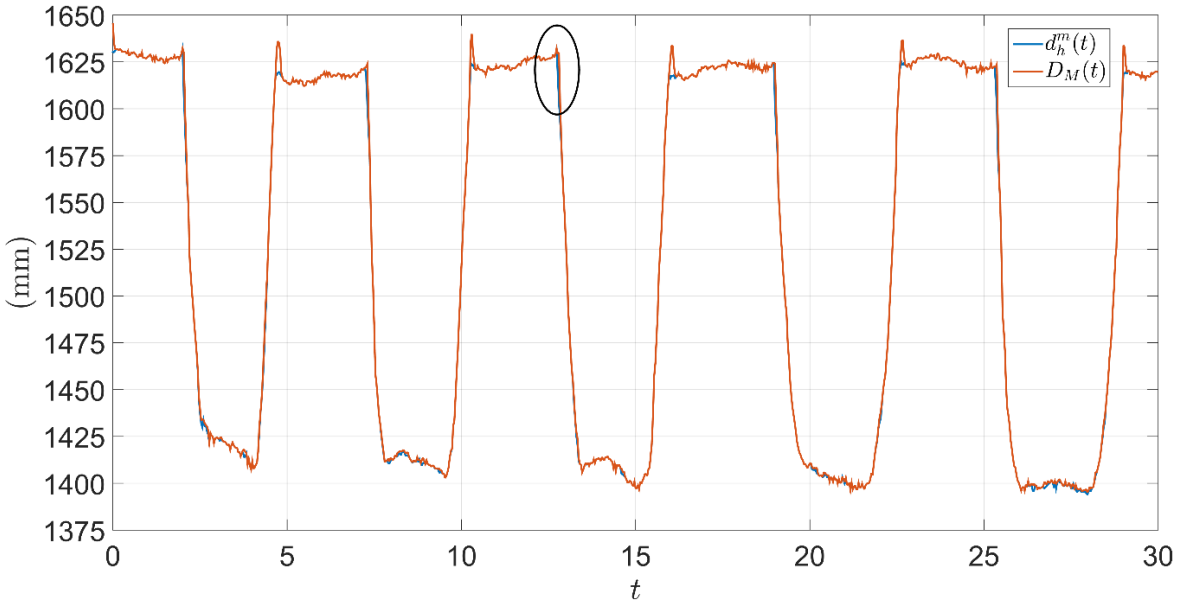

Figure 8. Signals d_h^m(t) (blue) and D_M (t) (red) obtained during the one-handed test.

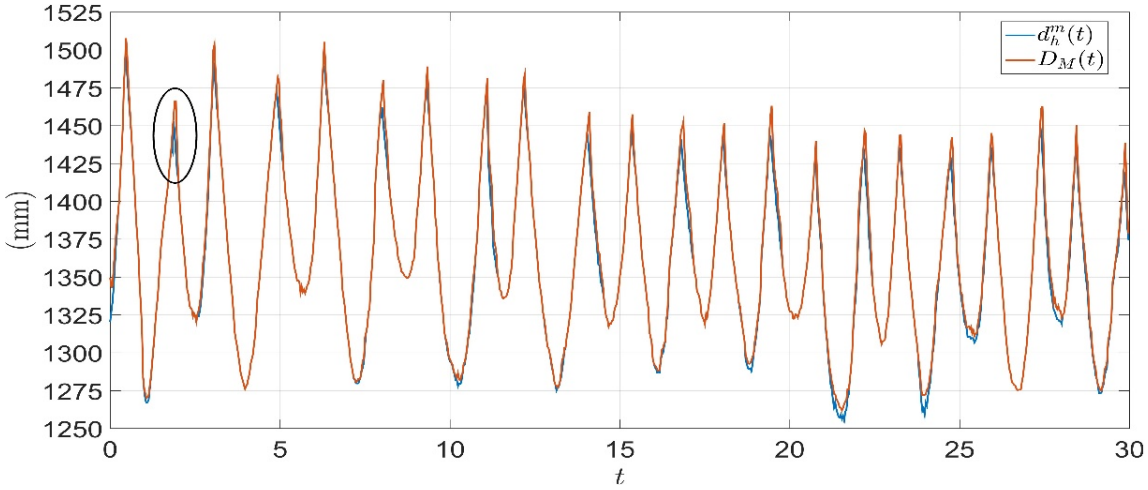

Figure 9. Signals d_h^m(t) (blue) and D_M(t) (red) obtained during the two-handed test.

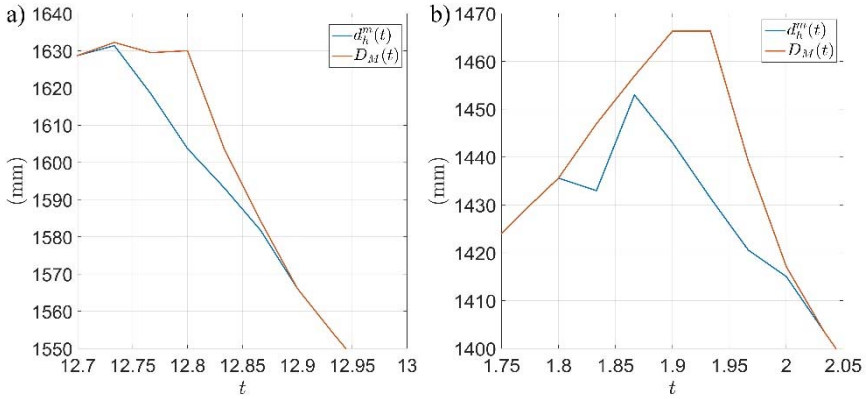

Figure 10. Zoom of the maximum error of the distance probability estimation (DPE) algorithm during the (**a**) one-handed test and (**b**) two-handed test.

4. Discussion

The experimental tests simulate the presence of six humans into the workspace. The tests show that the DPE algorithm was able to estimate the distance between human and manipulator with a maximum error of 26.29 mm (e_h = 3.52%) and 34.86 mm (e_h = 3.89%) for the one and two-handed test, respectively. The maximum relative errors are both smaller than 5%, in agreement with the specification used for tuning the parameter W. As considered in the worst-case scenario, the maximum estimation error made by the DPE algorithm arose when the next minimum distance point was not in the proximity square. In that case, the DPE algorithm entirely compensates these errors in less than 300 ms. As expected, the DPE algorithm presents a better performance in the one-handed test. It estimates the exact distance 79.78% of the time, while in the other test the distance is perfectly estimated the 53.44% of the time. This difference is caused by the D_h^m dynamics. Indeed, it can be observed in Figure 8 that during the one-handed test, the DPE algorithm is able to exactly estimate the minimum distance when the minimum distance signal is stationary (i.e., DPE algorithm most of the time founded the minimum distance in the proximity square). On the other hand, it can be observed in Figure 9 that the minimum point location discontinuities of the second test causes, more often, an inexact estimation of the minimum value.

Although the presented solution, where only one Microsoft Kinect depth sensor is used, can fail due to the shadow effects, it is worth noting that no information between depth cameras must be exchanged to compute the distances. Indeed, the usage of depth data information without the need of constructing any intermediate model makes our method intrinsically distributed. Therefore, the estimation of the distance between humans and manipulator surfaces can be performed without fusing information between depth cameras. This approach paves the way to the usage of multiple independent depth cameras for avoiding blinded zone without significantly affecting the computational costs.

5. Conclusions and Future Work

This work presents a novel distributed method to compute distances between humans and manipulator links. The proposed probabilistic approach directly searches the minimum distances on the depth information. The authors have demonstrated in this work that the algorithm allows for high frequency distance evaluation (i.e., 30 Hz) with e_h below 5% simulating six humans in the workspace. Additionally, the proposed algorithm is intrinsically distributed and modular. The modularity adds robustness to the overall system (a unit malfunctions does not interfere with the other units) and contributes to mitigate the problem of shadows by facilitating the incorporation of multiple points of view. Experimental tests show the DPE algorithm estimates the minimum distance with an RMS error of 5.93 mm (maximum error of 34.86 mm) in the worst-case scenario. Encouraged by the promising results, a dynamic optimization of the parameter W will be addressed for an autonomous enhanced performance of the algorithm. Finally, authors will evaluate the possibility to incorporate additional proximity squares for even more improving the robustness the algorithm in detecting any surrounding object. This solution can reduce the observed estimation errors when a spatial discontinuity of the minimum distance point occurs.

Author Contributions: Conceptualization: M.C., G.C., and A.D.; methodology: M.C. and A.D.; software: A.D. and I.V.; validation: M.C., A.D., I.V., and J.O.A.; data analysis: M.C., A.D., I.V., and J.O.A.; writing—original draft preparation: M.C., A.D., I.V., and J.O.A.; writing—review and editing: M.C., A.D., I.V., G.C., A.M., and J.O.A.; supervision: G.C. and A.M.; project administration: A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Community Seventh Framework Programme (FP7/20072013-FUTURA Project, G.A. 611963).

Acknowledgments: The authors wish to thank the FUTURA Consortium (www.futuraproject.eu/beneficiaries) and in particular M. Curti and M. De Micheli (Scienzia Machinale Srl, Pisa, Italy) for their help and valuable suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Johannsmeier, L.; Haddadin, S. A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes. *IEEE Robot. Autom. Lett.* **2016**, *2*, 41–48. [CrossRef]
2. Diodato, A.; Cafarelli, A.; Schiappacasse, A.; Tognarelli, S.; Ciuti, G.; Menciassi, A. Motion compensation with skin contact control for high intensity focused ultrasound surgery in moving organs. *Phys. Med. Boil.* **2018**, *63*, 035017. [CrossRef] [PubMed]
3. Haddadin, S.; Albu-Schaffer, A.; De Luca, A.; Hirzinger, G. Collision Detection and Reaction: A Contribution to Safe Physical Human-Robot Interaction. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 3356–3363.
4. Mazzocchi, T.; Diodato, A.; Ciuti, G.; De Micheli, D.M.; Menciassi, A. Smart sensorized polymeric skin for safe robot collision and environmental interaction. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 837–843.
5. O'Neill, J.; Lu, J.; Dockter, R.; Kowalewski, T. Practical, stretchable smart skin sensors for contact-aware robots in safe and collaborative interactions. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA, 26–30 May 2015; pp. 624–629.
6. Magrini, E.; De Luca, A. Hybrid force/velocity control for physical human-robot collaboration tasks. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 857–863.
7. Kasaei, S.H.; Sock, J.; Lopes, L.S.; Tomé, A.M.; Kim, T.-K. Perceiving, Learning, and Recognizing 3D Objects: An Approach to Cognitive Service Robots. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
8. Ragaglia, M.; Bascetta, L.; Rocco, P. Detecting, tracking and predicting human motion inside an industrial robotic cell using a map-based particle filtering strategy. In Proceedings of the 2015 International Conference on Advanced Robotics (ICAR), Washington, DC, USA, 26–30 May 2015; pp. 369–374.
9. Morato, C.; Kaipa, K.N.; Zhao, B.; Gupta, S.K. Toward Safe Human Robot Collaboration by Using Multiple Kinects Based Real-Time Human Tracking. *J. Comput. Inf. Sci. Eng.* **2014**, *14*, 011006. [CrossRef]
10. Kulic, D.; Croft, E. Safe planning for human-robot interaction. *J. Robot. Syst.* **2005**, *22*, 383–396. [CrossRef]
11. Safeea, M.; Mendes, N.; Neto, P. Minimum Distance Calculation for Safe Human Robot Interaction. *Procedia Manuf.* **2017**, *11*, 99–106. [CrossRef]
12. Phan, A.; Ferrie, F.P. Towards 3D human posture estimation using multiple kinects despite self-contacts. In Proceedings of the 2015 14th IAPR International Conference on Machine Vision Applications (MVA), Miraikan, Tokyo, Japan, 18–22 May 2015; pp. 567–571.
13. Cefalo, M.; Magrini, E.; Oriolo, G. Parallel collision check for sensor based real-time motion planning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1936–1943.
14. Dell'Acqua, F.; Fisher, R. Reconstruction of planar surfaces behind occlusions in range images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 569–575. [CrossRef]
15. Fabrizio, F.; De Luca, A. Real-Time Computation of Distance to Dynamic Obstacles with Multiple Depth Sensors. *IEEE Robot. Autom. Lett.* **2016**, *2*, 56–63. [CrossRef]
16. Flacco, F.; Kröger, T.; De Luca, A.; Khatib, O. A depth space approach to human-robot collision avoidance. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, St. Paul, MN, USA, 14–18 May 2012; pp. 338–345.
17. Yang, L.; Zhang, L.; Dong, H.; Alelaiwi, A.; El Saddik, A. Evaluating and Improving the Depth Accuracy of Kinect for Windows v2. *IEEE Sens. J.* **2015**, *15*, 1. [CrossRef]
18. Cosgun, A.; Bunger, M.; Christensen, H.I. Accuracy analysis of skeleton trackers for safety in HRI. In Proceedings of the Workshop on Safety and Comfort of Humanoid Coworker and Assistant (HUMANOIDS), Atlanta, GA, USA, 15 October 2013; pp. 15–17.
19. Boeing, A.; Braunl, T. Evaluation of real-time physics simulation systems. In Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality–TEEM 2017, Cádiz, Spain, 18–20 October 2007; p. 281.

20. Ong, C.J.; Gilbert, E. Fast versions of the Gilbert-Johnson-Keerthi distance algorithm: Additional results and comparisons. *IEEE Trans. Robot. Autom.* **2001**, *17*, 531–539.
21. Corke, P. A Simple and Systematic Approach to Assigning Denavit–Hartenberg Parameters. *IEEE Trans. Robot.* **2007**, *23*, 590–594. [CrossRef]
22. Xia, J.; Jiang, Z.; Liu, H.; Cai, H.; Wu, G. A Novel Hybrid Safety-Control Strategy for a Manipulator. *Int. J. Adv. Robot. Syst.* **2014**, *11*, 58. [CrossRef]