Generating Whole-Arm Avoidance Motion Through Localized Proximity Sensing

Simone Borelli, Francesco Giovinazzo, *Member, IEEE*, Alessandro Albini, *Member, IEEE*, Francesco Grella, *Member, IEEE*, and Giorgio Cannata, *Member, IEEE*

***Abstract*—This article presents a novel control algorithm for robotic manipulators in unstructured environments using proximity sensors partially distributed on the platform. The proposed approach exploits arrays of multizone Time-of-Flight (ToF) sensors to generate a sparse point cloud representation of the robot's surroundings. By employing computational geometry techniques, we fuse the knowledge of robot's geometric model with ToFs sensory feedback to generate whole-arm motion tasks. This allows to move both sensorized and nonsensorized links in response to unpredictable events such as moving obstacles. In particular, the proposed algorithm computes the pair of closest points between the environment cloud and the robot links, generating a reactive avoidance motion that is implemented as the highest priority task in a two-level hierarchical architecture. The algorithm allows the robot to move safely in unstructured environments, even without complete sensorization over the whole surface. Experimental validation demonstrates the algorithm's effectiveness in different operating scenarios, achieving comparable performances with respect to well established control techniques that rely on large area sensing architectures. We validate the approach on a human–robot interaction task involving 20 participants. In addition, we carry out a computational load analysis of the proposed algorithms. Our approach shows improvements in the distance margin between the robot and the environment up to 100 mm, due to the rendering of virtual avoidance tasks on nonsensorized links.**

***Index Terms*—Human–robot interaction, large-area sensors, prioritized control, proximity sensing.**

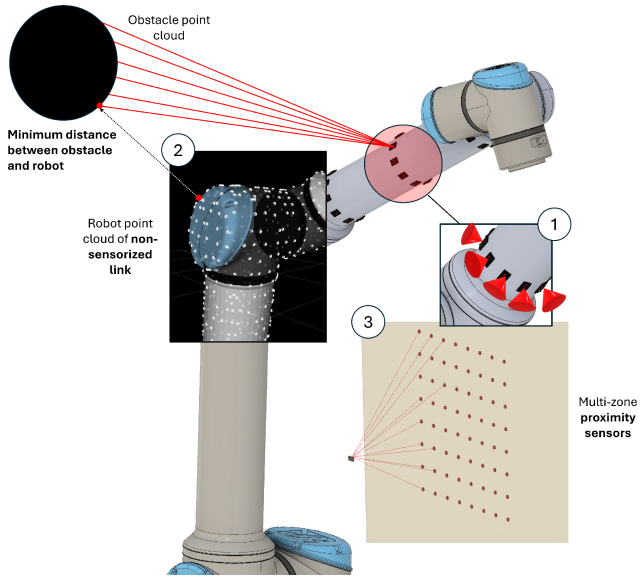


Fig. 1. 3-D model of a robotic manipulator partially covered with multizone proximity sensors (1). On the left side of the picture is highlighted an obstacle detected by one of the sensors. Our approach exploits the robot's geometric model (2) to define a dynamic avoidance task that acts on an arbitrary point over the robot's surface, regardless of its sensory layout. The representation of a typical output of multizone proximity sensors considered in this work is shown in (3), where most of the beams have been omitted for clarity.

I. Introduction

ROBOTIC manipulation in unstructured environments represents a significant challenge, as robots must operate safely and efficiently in the presence of dynamic and unpredictable obstacles. Traditional control approaches often rely on external sensing architectures based on cameras, tactile sensors, and more recently proximity sensing.

Visual-servoing architectures are a well-established approach relying on image processing techniques to perceive the surrounding space [1], [2], [3], [4], [5]. Vision-based systems, including cameras and motion capture setups, are used to enhance robot perception by monitoring the collaborative workspace, thus ensuring safety in human–robot interaction (HRI) applications [6], [7], [8], [9]. Current research has indicated that vision-based solutions perform well in critical situations. However, camera systems have important disadvantages in unstructured environments. In particular, their limited field of view can result in blind spots, leading to unexpected behavior in safety-critical scenarios due to temporary loss of information. Occlusions, which are highly dependent on camera positioning, can affect both eye-in-hand and eye-to-hand configurations.

Received 27 September 2024; revised 28 February 2025 and 24 April 2025; accepted 12 June 2025. Recommended by Technical Editor S. Pirozzi and Senior Editor S. Pirozzi. This work was supported by the SESTOSENSO (HORIZON EUROPE Research and Innovation Actions under Grant 101070310. *(Corresponding author: Francesco Grella.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the ethical committee for academic research (CERA) of the University of Genoa, Italy.

The authors are with the Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS), Università di Genova, 16145 Genova, Italy (e-mail: francesco.grella@edu.unige.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2025.3582432.

Digital Object Identifier 10.1109/TMECH.2025.3582432

In eye-in-hand setups, the field of view is constrained by the robot's pose and may be obstructed by its own movements or the manipulated object itself [10]. Eye-to-hand systems, meanwhile, are vulnerable to occlusions caused by moving obstacles or human operators, complicating the continuous tracking of the robot.

To address these issues, researchers have considered alternative or complementary sensing technologies. One of the most promising methods is large-area perception ([11], [12], [13], [14]), which relies on mounting distributed sensor networks on the manipulator's links, thereby enabling perception algorithms from the entire robot body.

Tactile perception is a preferred resource in tasks implying physical contact with the environment, such as autonomous clutter exploration [15], [16], [17], [18] and physical human–robot interaction (pHRI) [19]. In particular, large-area tactile sensing enables the integration of pressure sensors on complex curved surfaces, such as manipulator links, offering capabilities like multicontact detection and contact geometry reconstruction. Hence motion control algorithms that exploit distributed contact feedback allow robots to safely interact with the environment and gather information on its physical properties. Moreover, tactile sensing is insensitive to variations in light conditions and an extensive coverage with few blind spots allows to perceive contacts coming from any direction. Relying solely on touch when handling multiple sources of uncertainty is impractical and may result in potentially hazardous situations. For instance, in densely crowded environments or safety-critical applications like human–robot collaboration (HRC), tactile perception only provides local awareness of obstacles and cannot detect anything that is not already in contact with the robot. To overcome this limitation, the research community is extensively working on proximity sensing, both as a unique sensing modality [20], [21], [22] or as part of multimodal architectures [11], [23], [24], [25], [26].

Handling human presence within the robot's operational space is a well-established research topic which led to the definition of different approaches. Recent works described in [27] and [28] tackle the problem of safe HRI by adopting uncertainty-aware planning algorithms to estimate human motion. However, the approach described in [28] relies on a fixed perception setup based on Vicon cameras, which brings many limitations of camera-based approaches. In [27], the authors defined a task-dependent "danger zone" to trigger avoidance behavior and safely accomplish the task. Both methods are based on preestablished environmental hypotheses and involve first-time configuration procedures that depend on the structure of the workcell. In addition, these methods could limit the application flexibility due to their reliance on external sensing architectures. In this article, we propose an alternative approach based on large-area proximity sensors embodied on the manipulator to define dynamic reactive avoidance motion for operation in unstructured scenarios. Range-based approaches relying on LiDAR sensors have been recently presented in the context of human-crowded scenarios as depicted in [29]. LiDARs are a preferred choice when dealing with mobile robots, but can represent a limitation in collaborative manipulation tasks due to its fixed mounting and consequent occlusions.

Proximity-based whole-arm control enables the development of obstacle avoidance behaviors that can react to unexpected collisions without the limitations of camera-based algorithms. However, equipping the entire robot body with proximity sensors can be an expensive and complex engineering challenge. As a result, many solutions focus on integrating distributed sensors only on the final few links of the kinematic chain, as these are more prone to collisions [20], [30], [31].

Proximity servoing algorithms usually rely on distance measurements from sensor mounting locations to generate reactive collision avoidance tasks. However, these algorithms are often based on a limited set of control points–specifically, the sensor locations on the robot's body–to move the robot away from detected obstacles [21], [31], [32]. This approach has significant drawbacks in terms of flexibility, robustness, and fault tolerance.

Key limitations include that: 1) the manipulator body must be fully covered with proximity sensors to avoid blind spots; 2) in certain configurations, nonsensorized parts of the robot may be closer to the obstacle. As a result, while trying to avoid a collision with a sensorized link, the robot might unintentionally collide with a nonsensorized link; 3) if any sensor malfunction occurs, the obstacle avoidance behavior could become unreliable.

To address these limitations, this article presents a novel control algorithm that uses localized proximity sensing to enable reactive motion behaviors across the entire surface of the robot, including nonsensorized links. The algorithm employs a limited number of multizone proximity sensors, placed on the second link of a collaborative robot, to detect nearby obstacles. This sensing configuration is chosen to maximize ToFs coverage on the robot link most likely to collide during active operation and to prove the effectiveness of the implemented obstacle avoidance behavior on nonsensorized links. By processing this sparse sensor data using computational geometry techniques, it calculates the minimum distance between the environment and the robot's body. This approach extends reactive control to any arbitrary point on the robot links by generating motion tasks that are fed to a two-layer task priority architecture [33]. In particular, the obstacle avoidance behavior is given the highest priority, whereas the goal-reaching tasks are performed within the null space of the primary task. This design choice ensures an effective robot operation in complex, crowded and unstructured environments without affecting the safety-related objectives. In particular, the obstacle avoidance behavior is given the highest priority, whereas the goal-reaching tasks are performed within the null space of the primary task. This design choice ensures an effective robot operation in complex, crowded, and unstructured environments without affecting the safety-related objectives.

To the best of our knowledge, this is the first work addressing whole-arm obstacle avoidance with partial proximity sensorization. Moreover, providing nonsensorized links with reactive avoidance capabilites represents a considerable improvement in terms of sensory integration efforts and costs.

This article presents the following contributions.

1) A proximity data processing pipeline that efficiently filters out the robot model from the measurements of the proximity sensors.
2) An algorithm that extends sensor-based reactive control to the whole manipulator's surface without the need for a

complete sensorization.

3) An experimental evaluation of the proposed approach that considers static obstacles and human-robot interaction.

The rest of this article is organized as follows. Section II details the proposed methodology, focusing on the data processing algorithms and the implemented task priority control architecture. Section III describes the experiments carried out to validate our approach. Section IV showcases the obtained results and discusses the implications and limitations of our algorithm. Finally, Section V concludes this article and provides some insights for future research developments.

II. Methodology

The algorithm presented in this article relies on distance measurement, geometric information about the environment and the robot model to efficiently handle and react to approaching obstacles. However, a set of preprocessing operations must be performed on the raw data in order to ensure a proper identification of measurement corresponding to the robot body and to the objects close to the robot platform.

Initially, a spatial filtering operation is performed on the raw point cloud data to eliminate measurements outside the designated robot workspace. Although the selected proximity sensors can measure distances up to 4 m, the robot's operational range is limited to 1.3 m. As a result, subsequent computationally intensive algorithms, responsible for filtering the robot body measurements and computing the minimum distance point, process fewer data, thereby reducing the overall computational time.

A. Robot Body Measurement Filtering

Multiple proximity sensors are integrated all over the robot surface, as shown in Fig. 1. In specific joint configurations, ToF sensors Field of View (FoV) might intersect other links of the robotic platform, that would be treated as obstacles unless filtered out from the raw point cloud. To overcome such problems we developed an algorithm that processes the raw point cloud, eliminating all measurements corresponding to the robot body.

In the proposed solution, we assume to have N proximity sensors distributed on the second link of a robot manipulator, in the known locations $\mathbf{x}_i$, with $i = \{0, 1, \ldots, N\}$ as depicted in Fig. 1. Each sensor has a squared field-of-view and provides a set of measurements $\mathbf{d} = \{d_1, d_2, \ldots, d_M\}$ representing the absolute distance from the object with respect to the measuring device.

Distance measurements can be converted into a small point cloud P_i, by leveraging the sensor's geometric model as outlined in [25]. In particular, each of the 64 distance measurements d_i is represented as a point p_i in a 3-D coordinate system centered at the sensor's emitter, having its z-axis orthogonal to its surface. We use the trigonometric transformations described in (1), which are based on the manufacturer's specifications [34] consisting of the sets of angles α_i and β_i

$$p_i = \begin{cases} x_i = d_i \sin(\alpha_i) \\ y_i = d_i \tan(\beta_i) \\ z_i = d_i. \end{cases} \quad p_i \in \mathbb{R}^3, \ \forall\, i \in \{1, \ldots, 64\} \tag{1}$$

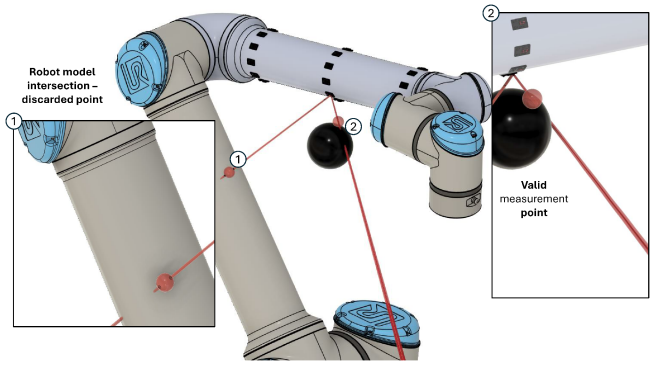


Fig. 2. Simplified 3-D description of the mesh removal algorithm. A ray is traced between a proximity sensor and all its sampled measurements, expressed as 3-D points. If the ray intersects the robot's model, the measurement is compared with the intersection point and if the distance is lower than a specified threshold, the measurement is discarded (1). The point (2) instead is an observation of an obstacle, and the corresponding ray does not intersect with the robot model, therefore the measurement is valid.

We also assume the full knowledge of the robot geometric model in a form of mesh composed of vertices and faces. In this respect we define $\mathcal{M}(\mathbf{q})$ the mesh where the vertex positions is computed depending on the joint configuration $\mathbf{q}$ with respect to the same reference frame used for each P_i.

Then, for each point $p \in P_i$ we check the intersection with the robot mesh $\mathcal{M}_i$. In particular, we compute a ray starting from the origin of the measurement $\mathbf{x}_i$ and passing through p. Then, to check for the intersection we leverage an AABB Tree data structure [35] that allows to perform intersection queries in logarithmic time.

The AABB tree $\mathcal{T}(\mathbf{q})$ is precomputed from the robot mesh $\mathcal{M}(\mathbf{q})$ and transformed for any given robot posture outside the algorithm shown in Algorithm 1. For each ray, we then check the intersection point between the ray and $\mathcal{T}(\mathbf{q})$.

In case of intersection, it is still necessary to check if the measurement point belongs to the robot mesh. In fact, the ray intersecting the robot body might also pass through an obstacle interposed between the sensor and the robotic platform, as shown in Fig. 2. In this case, we compute the distance between p and the intersection point with the robot body and compare it with a threshold ϵ, which accounts for the thickness of the 3-D printed covers mounted on the robot links. If the point p is beyond the threshold ϵ, it is added to the filtered environment cloud $\hat{P}$, otherwise it is removed.

The full algorithm is showed in Algorithm 1.

Algorithm 1 is executed at each time instant, for each number of links l. The time complexity for the removal of robot body measurements considers: I. The cost of the intersection check, which is logarithmic with respect to the number of vertices in the mesh V and proportional to the number of sensor measurements

Algorithm 1: Robot Body Measurement Removal.

Input : $P(\mathbf{q})$ and $\mathcal{T}(\mathbf{q})$ for $l = \{1, 2, \ldots, L\}$
Output: $\hat{P}$

$\hat{P} = [\,]$
for $i \leftarrow 1$ **to** L **do**
  foreach $p \in P_j$ **do**
    $\mathbf{r} = \texttt{computeRay}(\mathbf{p}, x_j)$
    $\mathbf{p}_\cap = \texttt{checkIntersection}(\mathcal{T}_i, \mathbf{r})$
    if $\mathbf{p}_\cap == NULL$ ***or*** $\texttt{computeDistance}(\mathbf{p}, \mathbf{p}_\cup) > \epsilon$ **then**
      $\hat{P} = \hat{P} \cup \mathbf{p}$
    end if
  end foreach
end for
return $\hat{P}$

Algorithm 2: Find Minimum Point.

Input : $\hat{P}$, $\mathcal{T}(\mathbf{q})$ for $l = \{1, 2, \ldots, L\}$
Output: mrp, mep, $\hat{l}$

$\mathbf{cmd} = MAX\ VALUE$
for $i \leftarrow 1$ **to** L **do**
  foreach $p \in \hat{P}$ **do**
    SD_i=$\texttt{squaredistance}(\mathrm{p}, \mathcal{T}_i)$
    if $SD_i < \mathbf{cmd}$ **then**
      $\mathbf{cmd} = SD_i$
      $\hat{l} = i$
      mrp=$\texttt{closestPoint}(mep, \hat{l})$
      mep=p
    end if
  end foreach
end for
return mrp, mep, $\hat{l}$

NM and the number of trees L; II. The cost of the distance computation between $\mathbf{p}$ and $\mathbf{p}_\cup$, which is constant. Therefore the overall cost is given by

$$\underbrace{O(LNM\log(V))}_{Intersection\ check} + \underbrace{O(LNM)}_{Distance\ computation} . \tag{2}$$

B. Minimum Distance Points

Once the robot body is filtered out, the resulting point cloud only provides information about the external environment, specifically the obstacles in proximity to the robot body. Then, it is necessary to identify the closest obstacle to the robot body in order to trigger an appropriate repulsive response.

The proposed algorithm finds, for each AABB tree $\mathcal{T}(\mathbf{q})$, the pair of points with the minimum squared distances, respectively, on the point cloud of the environment $\hat{P}$ and on the current AABB tree.

Once the iterations over $\mathcal{T}$ have ended, the function returns the point on the robot model mrp, the point in the environment cloud mep, and the corresponding link $\hat{l}$ that have the absolute minimum squared distance.

The full algorithm is showed in Algorithm 2.

The time complexity for the minimum distance point computation includes: 1) the number of iterations over the AABB trees, which is equal to the number of links l of the robot platform; the dimension of $\hat{P}$, proportional to NM; the time required to compute the squared distance from the current point of $\hat{P}$ and $\mathcal{T}(\mathbf{q})$, which is equal to $log(V)$; 2) the comparison performed to determine the current minimum, which is equal to LNM; 3) the time required to find the closest point on the robot, given the point from the point cloud p, which is in the average case equal to $\log(V)$.

Hence the actual cost is

$$\underbrace{O(LNM\log(V))}_{Minsquaredistancesearch} + \underbrace{O(LNM)}_{Currentmincomparison} + \underbrace{O(\log(V))}_{Closestpointsearch} . \tag{3}$$

The outcomes of the cloud processing pipeline are visualized in multiple configurations in Figs. 3 and 6.

C. Motion Control Algorithm

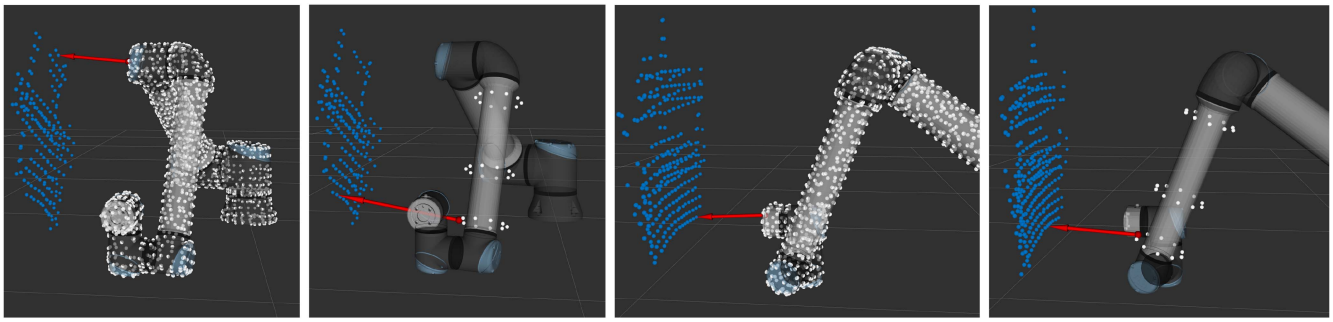
The control scheme implemented for this activity is based on the *Task Priority Control framework* presented in [36], which allows to satisfy several control objectives with different priorities. Specifically, in our application, we consider a robot controller with two priority layers. The highest priority controller deals with the safety of the overall system, implementing the obstacle avoidance behavior based on the preprocessed proximity sensors feedback. The lowest priority controller, instead, deals with the goal reaching task. The mathematical proof and formalism for the control architecture can be found in theoretical works [37] and applications [15], [38]. In this work, we have defined the reaching and avoidance tasks in a manner that adheres to the formal conditions of equality and inequality constraints typically considered in the literature, thereby ensuring consistency with established convergence criteria.

1) Safety Control Objective: The safety control task has the highest priority in the proposed two-layer control architecture. The goal is to increase the distance from the identified obstacles while trying to satisfy action-defining lower priority control objectives, such as reaching a desired goal position with the end-effector.

By defining $\dot{x}_o$ the velocity estimated for the obstacle closest to the robot body, $d_{\min} \in \Re$ the minimum distance that the robot should keep from the detected obstacle, Δ the constant offset used to create a buffer zone to smoothly activate the safety control task and $d_{curr} \in \Re$ the current minimum distance computed with Algorithm 2, we can write the feedback reference rate as

$$\dot{\tilde{x}} \triangleq \dot{x}_o + \lambda(d_{\min} + \Delta - d_{curr}), \quad \lambda > 0. \tag{4}$$

The continuous sigmoidal activation function $a^i(x)$ that regulates the activity of the safety control task $x > d_{\min}$ is described



(a) (b) (c) (d)

Fig. 3. Visual comparison of the whole-body minimum distance pair algorithm and its counterpart based on sensors mounting points. The white point cloud represents a downsampled model of the robot computed by its visual meshes (a) and (c) and proximity sensors locations in (b) and (d). The blue cloud represents an obstacle resembling our proposed experimental configurations. The red arrow is the vector that has the pair of closest points as its origin and tip.

by

$$a^i(x) = \begin{cases} 1 & \text{if } x < d_{\min} \\ s(x) & \text{if } d_{\min} < x < d_{\min} + \Delta \\ 0 & \text{if } x > d_{\min} + \Delta \end{cases} \tag{5}$$

such that $s(x)$ is any smooth, strictly decreasing function, Δ is the constant offset used to create a buffer zone, where the inequality $x > d_{\min}$ is already satisfied, but the activation value is still grater than zero, preventing chattering problems around the inequality control objective threshold $d_{\min}$. The task-induced Jacobian ${}^0J_{mrp/0}$, instead, is computed as

$${}^0J_{mrp/0} = {}^0S_{mrp/lmp}\ {}^0J_{lmp/0} \tag{6}$$

where S is the **rigid body Jacobian** defined as

$${}^0S_{mrp/lmp} \triangleq \begin{bmatrix} \mathbb{I}_{3\times3} & [{}^0r_{mrp/lmp}]^\top \\ 0_{3\times3} & \mathbb{I}_{3\times3} \end{bmatrix} \in \Re^{6\times6} \tag{7}$$

such that $[{}^0r_{mrp/lmp}]^\top$ is the skew symmetric distance between the point of minimum distance mrp and the frame of the link lmp to which the point belongs.

2) Goal-Reaching Control Objective: The goal reaching task is assigned the lowest priority and is executed within the null space of the primary, safety-related task. Specifically, this action-defining, equality control objective is used to reduce the position and orientation error between the robot end-effector and the desired goal frame. The feedback reference rate for the end-effector position control was computed as

$$\dot{\mathbf{x}}_{lin} \triangleq \lambda(\mathbf{x}_{goal} - \mathbf{x}_{ee})\,\lambda > 0 \tag{8}$$

where $\mathbf{x}_{goal} \in \Re^{3\times1}$ is the position of the goal, while $\mathbf{x}_{ee} \in \Re^{3\times1}$ is the current tool position, both computed with respect to the robot base.

The feedback reference rate for the end-effector orientation control was computed as

$$\dot{\mathbf{x}}_{ang} \triangleq \lambda \mathbf{n}\theta, \quad \lambda > 0 \tag{9}$$

where $\mathbf{n}\theta \in \Re^{3\times1}$ is the angle-axis representation of the orientation error.

The task-induced Jacobian for the goal reaching task was simply computed considering the linear and angular components of the robot end-effector Jacobian matrix

$${}^0J_{e/0} \triangleq \begin{bmatrix} {}^0J^L_{e/0} \\ {}^0J^A_{e/0} \end{bmatrix} \in \Re^{6\times6}. \tag{10}$$

The obtained task vectors and Jacobians are then exploited to compute suitable joint velocity references according with the control formulation described in [15]. The convergence of the proposed control algorithm is demonstrated in [36] and [37].

III. Experimental Validation

This section describes the tests performed to validate the implemented algorithms and control architecture. For this purpose, we used a 6-dof UR10e robotic arm by Universal Robotics, which was partially equipped with multizone ToF sensors. Specifically, VL53L8CX sensors, manufactured by ST Microelectronics [34], were integrated into flexible PCB arrays as part of the ProxySKIN technology [25]. Each proximity sensor generates an 8-by-8 matrix of distance measurements and is characterized by a field of view of 45°, represented in Fig. 1.

In this setup, three ring-shaped arrays, each consisting of 10 sensors, were mounted around the third link of the UR10e robot using custom 3-D printed covers, as shown in Fig. 4. This sensor placement is the result of a volumetric analysis that relates the FoV model of the sensors (obtained by manufacturing specifications) to their locations on the robot body [25]. The remaining robot surface was not sensorized to test the effectiveness of our approach.

The proposed algorithm was also validated on another robot platform, the Panda robot by Franka Emika, characterized by a more complex geometric model and 7-DoF. Three arrays of 10 ToF sensors were placed on different links of the Panda robot, specifically link 1, link 2, and link 7, as shown in Fig. 6. Performance tests on the robot body measurement filtering and on the minimum distance point computation were carried out, by varying two major performance metrics: 1) the robot model complexity; and 2) the size of the filtered environment cloud.

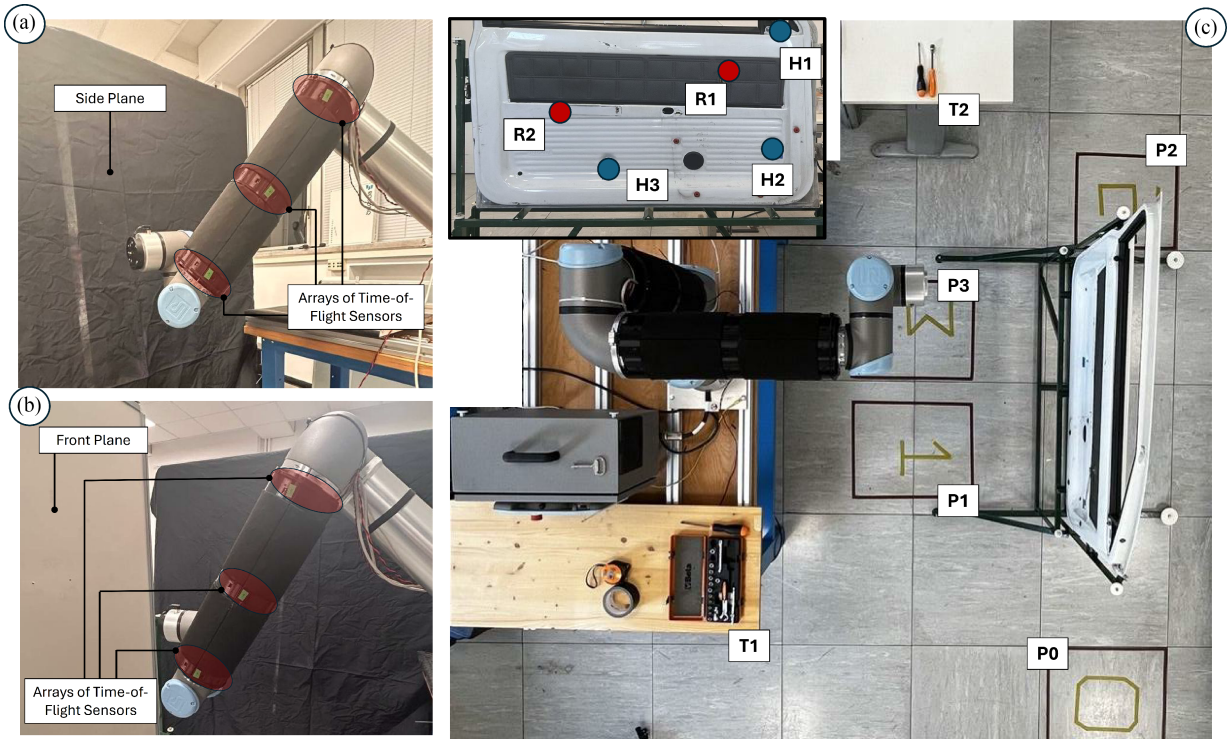


Fig. 4. Experimental setup configurations. (a) A planar object is placed besides the robot, to evaluate the algorithm's behavior on the uncovered links. (b) Another static planar obstacle is positioned in front of the robot, to evaluate algorithm's behavior on the uncovered flange. (c) Human–robot collaboration mockup. Robot motion waypoints R1 and R2 are marked with red dots, while blue dots represent the assembly locations for the subjects (H1, H2, and H3). The bird-eye view shows the locations used as markers for the assembly cycles progression, as well as the two toolbox locations represented as T1 and T2.

A. Algorithm Validation in Static Scenarios

The first set of experiments was conducted in a static environment, using fixed obstacles in proximity to the robot platform.

In the first trial, a planar surface was placed in front of the robot body [Fig. 4(a)], while in the second study it was placed on the right side of the UR10e [Fig. 4(b)]. In both cases, we defined a goal frame for the robot's end-effector beyond the planar surface and, by activating the obstacle avoidance safety control task, we checked the robot response to the following two different types of controllers.

1) A state of the art reactive controller, using the proximity sensors positions on the robot body as control points, presented in [31].
2) The controller explained in Section II, using any point of the robot model–including nonsensorized links–to enable reactive motion behaviors.

B. HRI Trials in Shared Workspace

The main validation task is focused on a car door inspection and assembly operation, involving the coexistence of the partially sensorized robot and a human operator within a small shared workspace. In this scenario, the operator is instructed to retrieve tools at fixed locations and perform predefined actions that resemble screwing operations on the inside of the car part, while the robot is continuously moving to reach specific waypoints within the workspace.

As illustrated in Fig. 4(c), the door is located in front of the robot, at a distance of 1.2 m from the robot base. Two waypoints (**R1** and **R2**) are marked on the car door to indicate the robot motion, while three different targets (**H1**, **H2**, and **H3**) are chosen for the operator to perform assembly tasks. Two additional locations are defined for tool retrieval (**T1** and **T2**), respectively, to the left and to the right of the robot base. Finally, four reference locations (**P1**, **P2**, **P3**, and **P4**) are marked on the ground for the human operator, corresponding to different phases of the the task progression. Variations in the toolbox position and assembly target points allow to observe the robot's performance under different environmental conditions.

The toolbox position is purposely chosen near the robot's base, requiring subjects to approach the first link, which are not sensorized. To validate our approach, we compare its performance against a baseline method described in [31], where only the sensor mounting points are used to move the robot away from obstacles. The performance of both algorithms is evaluated by measuring the distance between the robot's point cloud and the closest point of the environment's point cloud over time.

The HRI trials involved 20 participants, aged 24 to 40, with heterogeneous professional backgrounds. For each attendee, two different assembly tests were performed in close proximity of the moving robot using the baseline algorithm and the proposed whole-arm controller as follows.

1) In the first trial, subjects were instructed to approach the shared workspace starting from position **P0** and

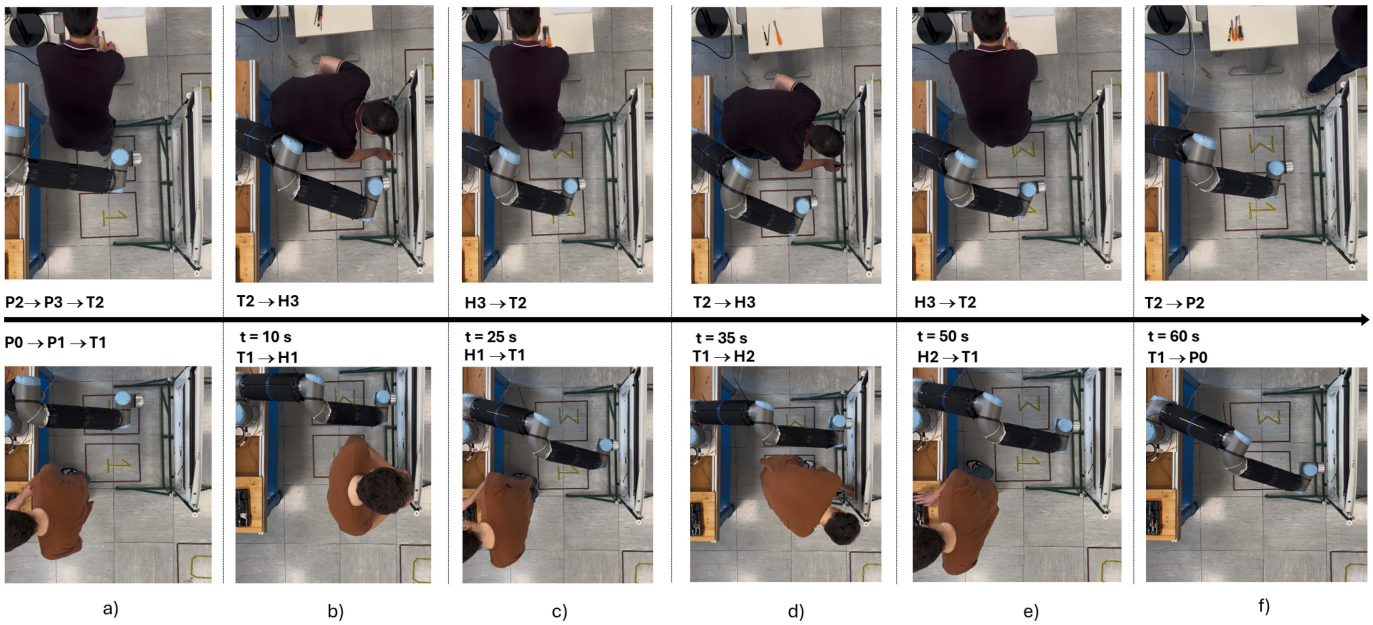


Fig. 5. Timeline of the assembly cycle. (a) The subject enters the shared workspace and reaches the target toolbox location. (b) When the first acoustic signal is given, the subject moves to the first assembly target, while the robot attempts to reach its assigned waypoints. (c) The subject is signaled to move back to the tool location to simulate a tool change. (d) After 35 seconds in the cycle, the subject moves towards the second assembly location and performs the mockup task for 15 seconds more. (e) At the 50 seconds mark the subject is prompted to reach the toolbox one last time to leave the tools, then is asked to leave the workspace and to go back to the initial location (**P0**/**P2**) as shown in (f).

proceeding to position **P1**, located halfway between the toolbox location **T1** and the **H1** and **H2** markers on the car door. The assembly cycle required the subjects to move between the toolbox and the car door in proximity to the ground marker **P1**. In particular, they were asked to move from the toolbox **T1** to the car door, perform the assembly operation in **H1**, change the tool back in **T1**, assemble the second component in **H2**, put down the tool in **T1** and leave the shared workspace. The assembly phases were dictated by a timer-based acoustic signal, notifying a change of action at predefined time intervals.

2) The second trial had the very same structure of the first, except for the different task locations. Users were asked to move from **P2** to **P3**, retrieve the tool in **T2** and simulate an assembly operation at **H3** twice, changing the tool in **T2** before the second assembly operation.

In Fig. 5 are illustrated the assembly steps performed by two subjects during the first (top) and in the second (bottom) assembly cycle over time.

C. Computational Analysis and Generalization

To demonstrate the generalization of the implemented algorithms across different robot platforms, we conducted a performance analysis of their computational complexity. Two robot platforms were considered: a UR10e and a Panda robot, both equipped with three arrays of 10 ToF sensors each. Specifically, we performed an execution time analysis of Algorithms 1 and 2 by varying the following parameters.

1) *Geometric model resolution:* The number of vertices per mesh was varied from 10^2 to 10^4 points.

2) *Environment point cloud size:* Three different configurations were considered: a) the raw environment cloud, containing more than 1500 data points; b) a mildly spatially filtered cloud, including obstacles within 2 m of the robot body, totaling approximately 600 data points; c) an extensively spatially filtered cloud, including obstacles within 1 m of the robot body, totaling approximately 200 data points.

Data were collected with the robot in a static configuration by placing an obstacle in close proximity to its body while running the perception stack used in the HRI trials.

In Fig. 6 are illustrated the real Panda robot setup, used for the computational analyses of the implemented algorithms, and three virtual 3-D representations characterized by different robot model resolutions and spatial filtering configurations.

IV. Results and Discussion

In this section we present quantitative results obtained from our experimental trials, providing a comparison between our proposed whole-body algorithm **(WB)**and the baseline state-of-the-art approach based on motion control of the sensors' mounting points **(SM)**.

A. Results of Static Obstacles Trials

For this set of experiments, we evaluated the minimum distance between the robot and the environment point cloud while performing a 6-dof reaching task in the presence of a static planar obstacle as described in Section III-A. We evaluated the norm of the minimum distance between the point clouds

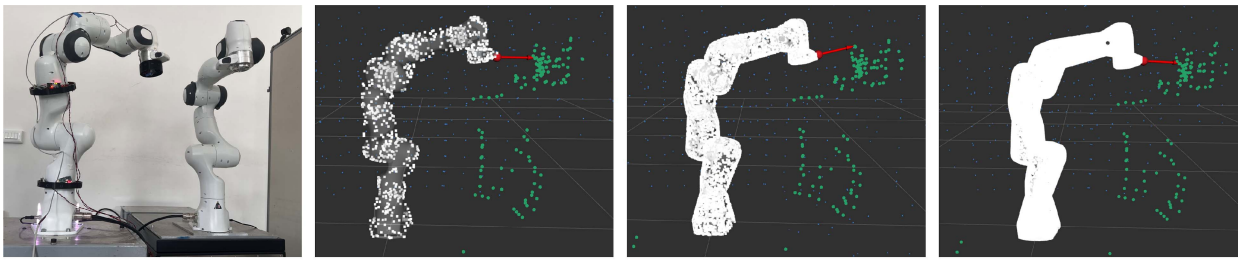


Fig. 6. Generalization of the computational load analysis to other robot platforms. (a) Real experimental setup, showing a sensorized Panda robot (on the left) and a static obstacle (robot on the right). (b)–(d) Virtual representation of the experimental setup considering different resolutions of the robot link models (white point cloud). The green point cloud represents the filtered measurements close to the robot body. The blue point cloud represents the raw sensors data. The red arrow represent the vector that has the pair of closest points as its origin and tip. The robot link geometric models feature respectively: (b) 100 points, (c) 1000 points and (d) 10000 points.

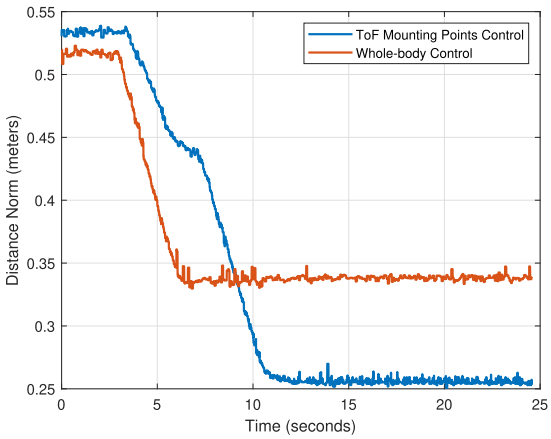


Fig. 7. Comparative analysis between the presented whole-body algorithm and the baseline during the front plane test.

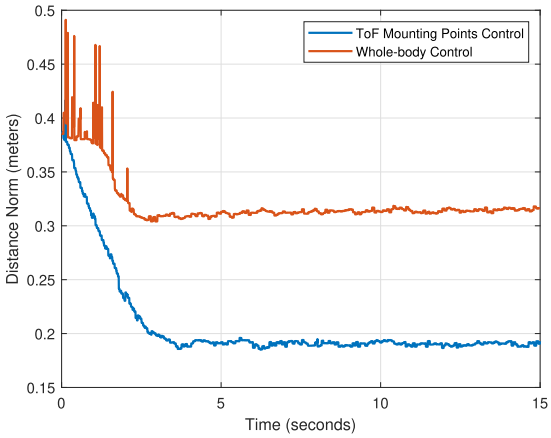


Fig. 8. Comparative analysis between the presented whole-body algorithm and the baseline during the side plane test.

representing the environment and the robot. The outcome of the trial performed with the front panel is shown in Fig. 7: when the whole-body controller (**WB**) is used, the robot is able to keep a greater distance with respect to the baseline method, using sensor mounting positions (**SM**) to drive away the robot from detected obstacles. The robot exhibits this behavior because with **WB**the control point is located on the flange, which is in fact the link closest to the obstacle.

A similar result (Fig. 8) is obtained considering a static obstacle on the left side of the robot's workstation, as shown in Fig. 4(a)–(b). As for the previous case, our approach allows the robot to maintain a larger safety distance from the objects detected.

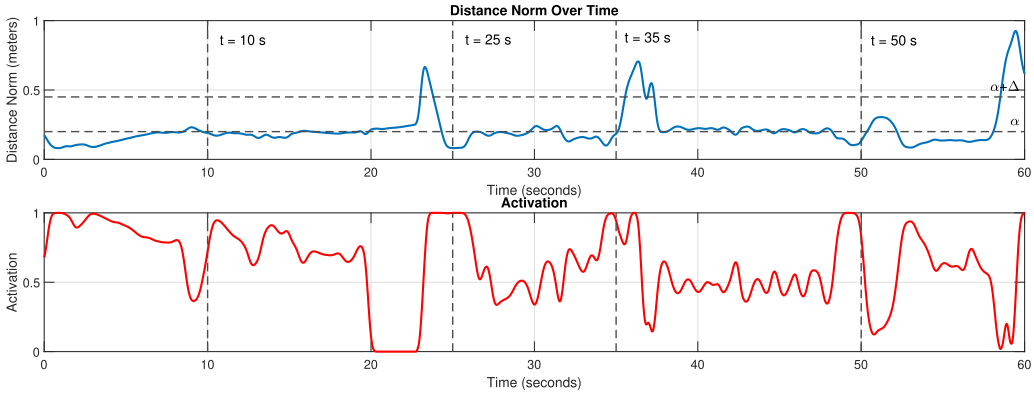
TABLE I
MEASURED DISTANCES AND THEIR CORRESPONDING MAXIMUM STANDARD DEVIATIONS FOR THE WHOLE ASSEMBLY CYCLES

	$\overline{d}$(m)	σ_{max}(m)
SM Assembly Task 1	0.331	0.129
WB Assembly Task 1	0.339	0.126
SM Assembly Task 2	0.285	0.189
WB Assembly Task 2	0.310	0.167

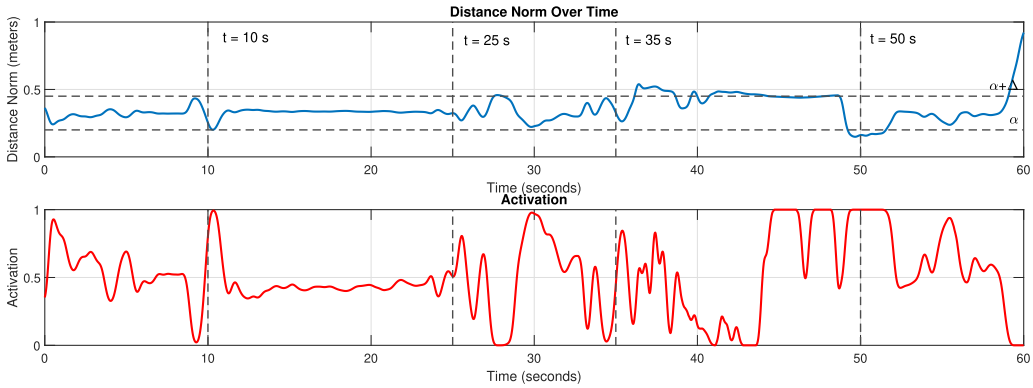
B. Results of HRI Trials

In this set of experiments, we observed the robot's behavior in the presence of humans moving inside the shared worskpace, as described in Section III-B. We asked 20 different subjects to perform two assembly cycles, starting, respectively, from position **P0** and position **P2**, as the robot was controlled using the **WB**and the **SM**approaches. The **SM**and **WB**algorithms were compared by analyzing the minimum distance between the robot and the environment, both represented as point clouds. In particular, we evaluated the average value of the minimum distance $\overline{d}$ and the maximum value of its standard deviation $\sigma_{\max}$ over each trial. The results are summarized in Table I. Moreover, we performed an in-depth analysis of the most critical stages of the assembly task, represented by **H1**, **H2**, and **H3** to better assess the robot's behavior when the subjects were standing close to the arm and were interfering with the robot's path. The results are summarized in Table II.

The statistic analysis on the average minimum distance along the whole cycles shows that when the operator is approaching the workcell from **P0** (Task 1) the sensors' mounting locations are generally as close as the nonsensorized robot wrist, explaining why the **WB**trials show a comparable performance to the



(a)



(b)

Fig. 9. Time-based plots of two complete HRI trials. (a) shows data related to a trial performed with the baseline algorithm (SM), having the subject approaching the workspace from P2. (b) shows the outcome of a trial performed with the novel whole-body approach. It is worth noticing that the WB algorithm allows to better constrain the motion to stay within assigned safety thresholds.

TABLE II
MEASURED DISTANCES AND THEIR CORRESPONDING MAXIMUM STANDARD DEVIATIONS FOR CRITICAL ASSEMBLY OPERATIONS

	$\overline{d}$(m)	σ_{max}(m)
SM - Assembly Operation on H1	0.312	0.099
WB - Assembly Operation on H1	0.327	0.111
SM - Assembly Operation on H2	0.305	0.143
WB - Assembly Operation on H2	0.316	0.092
SM - Assembly Operation on H3 (first run)	0.224	0.130
WB - Assembly Operation on H3 (first run)	0.249	0.103
SM - Assembly Operation on H3 (second run)	0.245	0.140
WB - Assembly Operation on H3 (second run)	0.257	0.122

SMones. In contrast, when the assembly task starts from **P2** (Task 2) the **WB**algorithm exhibits performance improvements with respect to **SM**, since the avoidance tasks are generated on the nonsensorized robot wrist, which gets closer to the subject when using the **SM**controller.

The results obtained by evaluating the mean $\overline{d}$ and maximum standard deviations σ_{max} of the minimum distance during the critical assembly phases **H1**, **H2**, and **H3** follow the same trend described in Table I. In particular, for all the above cases, the proposed **WB**controller achieves slightly better results in terms of minimum safety distance compared to the baseline **SM**algorithm, while the maximum standard deviations have similar trends.

Fig. 9 shows the minimum distance and the avoidance activation function over two trials performed in **SM - Task 1**and **WB - Task 2**conditions, respectively. Vertical markers indicate the time intervals at which the operator moves within the workspace. Horizontal lines in the minimum distance plots represent the thresholds that define the transition region of the activation function. It is clearly visible how the operator's movements cause a sudden increase in the activation in both experiments. Moreover, the minimum distance plots distinctly show how the **WB**method helps keeping an overall larger distance margin from the operator, while the **SM**algorithm struggles to maintain a minimum distance threshold.

C. Results of the Algorithms Computational Analysis

In this test, we focused on the computational cost of the algorithms implementing the robot body measurement filtering and the evaluation of minimum distance points. As described in Section III-C, we considered two different robot platforms, a UR10e and a Panda robot, characterized by different geometric structures. Tests were carried out on a workstation running Linux Ubuntu 22.04, mounting an Intel i7-12th Gen. We considered

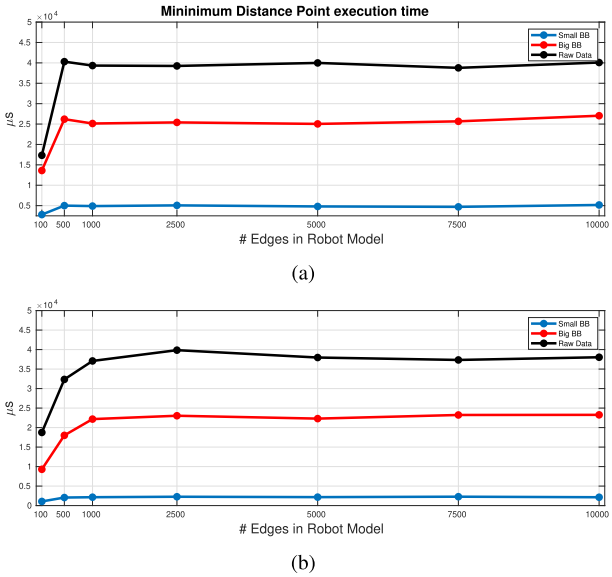


Fig. 10. Analysis of the computational complexity for the minimum distance point evaluation (Algorithm 2). (a) Results obtained for the Panda Robot; (b) Results obtained for the UR10e robot.

variations of two performance-affecting parameters: the number of points of the robot geometric model, varying from 10^2 to 10^4, and the size of the environment point cloud, varying from less than 200 points to more than 1500.

Figs. 9 and 10 show the results obtained by comparing the UR10e and the Panda robot performances for Algorithm 1 and 2, respectively.

The results obtained for the Panda and the UR10e robot platforms are comparable and consistent with the theoretical computational costs outlined in Sections II-A and II-B, characterized by a logarithmic trend. In particular, by increasing the number of points in the robot geometric model, the time required for filtering the robot body measurements and finding the minimum closest points reach a horizontal asymptote that is proportional to the number of points of the environment point cloud. As the number of points in the environment point cloud increases, so does the computational cost.

The computational cost of Algorithm 2 has an order of magnitude greater than Algorithm 1 because, as noted in (3), it depends on an additional logarithmic term that deals with the search of minimum distance points.

The results obtained from the computational analysis are acceptable for our application, given that the robot is controlled at low speeds in the presence of fixed obstacles and humans. Even in the worst-case scenario–10^4 points per robot link model and $1.5 \cdot 10^3$ points of the raw environment cloud–both algorithms perform their computations within the proximity sensors cycle time, set to 66.7 ms. However, to boost the performances of Algorithms 1 and 2, it is reasonable to use a coarser representation for the robot geometric model and to perform a prior spatial filtering of the environment point clouds.

Finally, the computational analysis on the UR10e and the Panda robot suggest that Algorithm 1 and 2 can be generalized to more complex robotic platforms.

V. Conclusion

In this article, we presented a novel proximity servoing algorithm that generates reactive collision avoidance behavior by exploiting a limited set of multizone range sensors integrated on a robotic manipulator. In particular, we propose a methodology that leverages the geometric model of the robot to find its closest point with respect to the environment, represented as a point cloud. We compared our method with a baseline control algorithm for reactive avoidance [31], showing comparable qualitative and quantitative performances. Moreover, the presented algorithm can exploit any arbitrary point on the robot's surface to perform avoidance motion, showing improvements in the distance margin due to the definition of control tasks on nonsensorized links. To the best of our knowledge, this is the first article addressing whole-arm obstacle avoidance with partial robot sensorization. Our approach represents an improvement in terms of integration efforts, since it allows to endow a robot with proximity perception without the need of sensorizing its entire surface.

It is worth noting that while the proposed approach effectively addresses common limitations of vision-based sensing, it is not entirely immune to blind spots, which are primarily due to the placement of sensors on the robot links. During the HRI experimental sessions, two failure cases were observed in which the robot approached the human subject too closely and had to be stopped manually. A post-experiment analysis of the raw environment point cloud confirmed that these instances occurred in specific robot configurations, particularly when the human subject entered the shared workspace from a region not monitored by the proximity sensors. This limitation can be mitigated through a detailed analysis of blind spots to optimize sensor placement. In addition, strategies that leverage the robot's motion to improve coverage of previously unobserved areas are currently under investigation and will be explored in future work.

References

[1] J. Kim and Y. Do, "Moving obstacle avoidance of a mobile robot using a single camera," *Procedia Eng.*, vol. 41, pp. 911–916, Jan. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705812026628

[2] R. Lagisetty, N. K. Philip, R. Padhi, and M. S. Bhat, "Object detection and obstacle avoidance for mobile robot using stereo camera," in *Proc. 2013 IEEE Int. Conf. Control Appl.*, 2013, pp. 605–610. [Online]. Available: https://ieeexplore.ieee.org/document/6662816

[3] L. S. Scimmi, M. Melchiorre, S. Mauro, and S. P. Pastorelli, "Implementing a vision-based collision avoidance algorithm on a UR3 robot," in *Proc. 23rd Int. Conf. Mechatron. Technol.*, Oct. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8932105

[4] H. Nascimento, M. Mujica, and M. Benoussaad, "Collision avoidance in human-robot interaction using kinect vision system combined with robot's model and data," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10293–10298.

[5] H. Liu and L. Wang, "Collision-free human-robot collaboration based on context awareness," *Robot. Comput.- Integr. Manuf.*, vol. 67, Feb. 2021, Art. no. 101997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584520302088

[6] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco, "Prediction of human activity patterns for HumanRobot collaborative assembly tasks," *IEEE Trans. Ind. Inform.*, vol. 15, no. 7, pp. 3934–3942, Jul. 2019.
[7] L. Bascetta et al., "Towards safe human-robot interaction in robotic cells: An approach based on visual tracking and intention estimation," in *Proc. 2011 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, 2011, pp. 2971–2978.
[8] A. Tellaeche, I. Maurtua, and A. Ibarguren, "Human robot interaction in industrial robotics. Examples from research centers to industry," in *Proc. 2015 IEEE 20th Conf. Emerg. Technol. Factory Automat.*, 2015, pp. 1–6.
[9] G. Du and P. Zhang, "A markerless HumanRobot interface using particle filter and Kalman filter for dual robots," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2257–2264, Apr. 2015.
[10] K. He et al., "Visibility maximization controller for robotic manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8479–8486, Jul. 2022.
[11] E. Y. P. Mittendorfer and G. Cheng, "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot," *Adv. Robot.*, vol. 29, no. 1, pp. 51–67, 2015. [Online]. Available: https://doi.org/10.1080/01691864.2014.952493
[12] Y. Zhou, J. Zhao, P. Lu, Z. Wang, and B. He, "TacSuit: A wearable large-area, bioinspired multimodal tactile skin for collaborative robots," *IEEE Trans. Ind. Electron.*, vol. 71, no. 2, pp. 1708–1717, Feb. 2024.
[13] P. Maiolino, M. Maggiali, G. Cannata, G. Metta, and L. Natale, "A flexible and robust large scale capacitive tactile system for robots," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3910–3917, Oct. 2013.
[14] M. J. Yang, K. Park, W. D. Kim, and J. Kim, "Robotic skin mimicking human skin layer and pacinian corpuscle for social interaction," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 4, pp. 2709–2719, Aug. 2024, [Online]. Available: https://ieeexplore.ieee.org/document/10360302
[15] A. Albini, F. Grella, P. Maiolino, and G. Cannata, "Exploiting distributed tactile sensors to drive a robot arm through obstacles," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4361–4368, Jul. 2021.
[16] Z. Ye et al., "Soft robot skin with conformal adaptability for on-body tactile perception of collaborative robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5127–5134, Apr. 2022.
[17] D. Brouwer et al., "Tactile-informed action primitives mitigate jamming in dense clutter," in *Proc. 2024 IEEE Int. Conf. Robot. Automat.*, 2024, pp. 7991–7997.
[18] S. Jiang and L. L. Wong, "A hierarchical framework for robot safety using whole-body tactile sensors," in *Proc. 2024 IEEE Int. Conf. Robot. Automat.*, 2024, pp. 8021–8028.
[19] F. Grella, A. Albini, and G. Cannata, "Voluntary interaction detection for safe human-robot collaboration," in *Proc. 6th IEEE Int. Conf. Robotic Comput.*, 2022, pp. 353–359.
[20] S. Tsuji and T. Kohama, "Proximity skin sensor using time-of-flight sensor for human collaborative robot," *IEEE Sensors J.*, vol. 19, no. 14, pp. 5859–5864, Jul. 2019.
[21] Y. Ding and U. Thomas, "Collision avoidance with proximity servoing for redundant serial robot manipulators," in *Proc. 2020 IEEE Int. Conf. Robot. Automat.*, 2020, pp. 10249–10255.
[22] S. Kumar, S. Arora, and F. Sahin, "Speed and separation monitoring using on-robot time-of-flight laser-ranging sensor arrays," in *Proc. IEEE 15th Int. Conf. Automat. Sci. Eng.*, 2019, pp. 1684–1691.
[23] D. Hughes, J. Lammie, and N. Correll, "A robotic skin for collision avoidance and affective touch recognition," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1386–1393, Jul. 2018.
[24] C. Abah, A. L. Orekhov, G. L. Johnston, P. Yin, H. Choset, and N. Simaan, "A multi-modal sensor array for safe human-robot interaction and mapping," in *Proc. 2019 Int. Conf. Robot. Automat.*, Montreal, QC, Canada, 2019, pp. 3768–3774.
[25] F. Giovinazzo, F. Grella, M. Sartore, M. Adami, R. Galletti, and G. Cannata, "From cysKIN to proxySKIN: Design, implementation and testing of a multi-modal robotic skin for human-robot interaction," *Sensors*, vol. 24, no. 4, 2024, Art. no. 1334. [Online]. Available: https://www.mdpi.com/1424-8220/24/4/1334
[26] S.-H. Heo and H.-S. Park, "Proximity perception-based grasping intelligence: Toward the seamless control of a dexterous prosthetic hand," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 3, pp. 2079–2090, Jun. 2024, [Online]. Available: https://ieeexplore.ieee.org/document/10301192
[27] M. Li, J. Qin, Z. Wang, Q. Liu, Y. Shi, and Y. Wang, "Optimal motion planning under dynamic risk region for safe human–robot cooperation," *IEEE/ASME Trans. Mechatron.*, vol. 30, no. 2, pp. 1050–1060, Apr. 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10584116
[28] W. Liu, K. Eltouny, S. Tian, X. Liang, and M. Zheng, "Integrating uncertainty-aware human motion prediction into graph-based manipulator motion planning," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 4, pp. 3128–3136, Aug. 2024.
[29] K. Kudo, N. Kawaguchi, M. Adachi, K. Sekiguchi, and K. Nonaka, "LiDAR-Based pedestrian flow estimation and its application to a self-driving electric wheelchair," in *Proc. 2024 IEEE Int. Conf. Adv. Intell. Mechatron.*, 2024, pp. 1061–1067. [Online]. Available: https://ieeexplore.ieee.org/document/10636953/?arnumber=10636953
[30] S. E. Navarro, S. Koch, and B. Hein, "3D contour following for a cylindrical end-effector using capacitive proximity sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 82–89.
[31] G. Caroleo, F. Giovinazzo, A. Albini, F. Grella, G. Canata, and M. Perla, "A proxy-tactile reactive control for robots moving in clutter," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 733–739. [Online]. Available: https://ora.ox.ac.uk/objects/uuid:137e5c69-7a0d-4711-86f2-d86cc0884166
[32] G. B. Avanzini, N. M. Ceriani, A. M. Zanchettin, P. Rocco, and L. Bascetta, "Safety control of industrial robots based on a distributed distance sensor," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 6, pp. 2127–2140, Nov. 2014.
[33] A. Albini, S. Denei, and G. Cannata, "Human hand recognition from robotic skin measurements in human-robot physical interactions," in *Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, 2017, pp. 4348–4353.
[34] "Datasheet vl53l8cx - low-power high-performance 8x8 multizone time-of-flight sensor (TOF)," 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/vl53l8cx.pdf
[35] "Fast intersection and distance computation (AABB tree)," 2023. [Online]. Available: https://doc.cgal.org/latest/AABB_tree/index.html
[36] E. Simetti, G. Casalino, F. Wanderlingh, and M. Aicardi, "Task priority control of underwater intervention systems: Theory and applications," *Ocean Eng.*, vol. 164, pp. 40–54, 2018.
[37] S. B. Slotine and B. Siciliano, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. Adv. Robot.*, 1991, pp. 1211–1216.
[38] E. Simetti, G. Casalino, S. Torelli, A. Sperinde, and A. Turetta, "Floating underwater manipulation: Developed control methodology and experimental validation within the trident project," *J. Field Robot.*, vol. 31, no. 3, pp. 364–385, 2014.

Simone Borelli is currently working toward the Ph.D. degree in robotics and autonomous systems as a member of the Mechatronics and Automatic Control Laboratory (MACLab), University of Genoa, Genoa, Italy.

He is an active member of the SESTOSENSO project and his research interests include robotics, robot sensing and control systems.

Francesco Giovinazzo (Member, IEEE) is currently working toward the Ph.D. degree in robotics and autonomous systems as a member of the Mechatronics and Automatic Control Laboratory (MACLab), University of Genoa, Genoa, Italy.

He is currently working on reactive robot control based on distributed tactile and proximity sensors for safe human–robot interaction.

Mr. Giovinazzo's master thesis project was awarded as "Best thesis of 2021" from the "Comitato Elettrotecnico Italiano (CEI)" in 2021.

Alessandro Albini (Member, IEEE) received the Ph.D. degree in robotics and autonomous systems from the University of Genoa, Genoa, Italy, in 2020.

He joined the Soft Robotics Lab as a Post Doctoral Researcher in 2020. He is currently working on robotic tactile perception under the supervision of Professor Perla Maiolino. His research interests include artificial robot skin technologies and their applications to human–robot interaction and robot control.

Francesco Grella (Member, IEEE) received the master's degree in robotics engineering and the Ph.D. degree in Robotics and Autonomous Systems from the University of Genoa, Genoa, Italy, in 2019 and 2024, respectively.

He is a Postdoctoral Research Fellow with the University of Genoa. His research interests include sensor-based robot control, tactile-based robotic perception, and tactile sensor design and fabrication.

Giorgio Cannata (Member, IEEE) received the Laurea degree in electronic engineering from the University of Genoa, Genoa, Italy, in 1988.

He is a Full Professor of automatic control with the Polytechnic School of Engineering, University of Genoa. He is the Scientific Coordinator of the Mechatronics and Automatic Control Laboratory (MACLab), University of Genoa. His current research interests include robotics, mechatronics, and robot sensing and control systems.