Real-Time Object Detection for Collaborative Robots in Close Proximity using DBSCAN in Super Sparse Voxel Environments

Anas Atmani[1], Philipp Lüdtke[1], Sophie Charlotte Keunecke[2], Burkhard Corves[2]
[1]*Faculty of Mechanical Engineering*, RWTH Aachen University
[2]*Institute of Mechanism Theory, Machine Dynamics and Robotics*, RWTH Aachen University

***Abstract*—The paradigm shift toward Industry 5.0 requires a fundamental change in human-robot collaboration. This change involves transitioning from reactive collision detection to proactive, human-centric safety frameworks. Although extrinsic sensing systems and resource-intensive AI-based perception and decision-making pipelines provide rich environmental context in complex scenarios, occlusion and significant processing latencies. This research addresses these challenges by presenting a low-latency, intrinsic, 3D motion tracking system that uses a Time-of-Flight sensor ring mounted directly on a Cobot link. The core contribution is a data processing pipeline integrating a hysteresis-based temporal occupancy filter that eliminates ghosting artifacts and spatial merging. Exploiting the discrete topology of super-sparse voxel grids, we derive an optimized DBSCAN clustering configuration that ensures feature continuity while maintaining real-time performance. This approach minimizes geometric detail to the level strictly necessary for robust tracking, thereby maximizing computational performance. Experimental validation in high-proximity scenarios, including human-to-human interactions, demonstrates stable tracking at internal frequencies of 14–16 Hz. The findings confirm that combining low-cost onboard sensing with a zero-shot sparse voxel pipeline yields predictable trajectories for collision avoidance, thus facilitating the inclusivity and safety requirements of modern industrial code.**
The code is available at Predictive Collision Avoidance Repository.

***Index Terms*—Time-of-Flight (ToF) Sensors; Voxelization; Object Detection; Human-Robot Collaboration; Real-Time Robot Control; Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**

I. Introduction

Industry 5.0 (I 5.0) signifies a pivotal realignment of industrial priorities, emphasizing human-centric collaboration, sustainability, and resilience rather than the pursuit of automation for its own merit [1]. It builds on the digital and cyber-physical foundations of previous industrial advancements, particularly Industry 4.0 (I 4.0). I 5.0 utilizes technologies such as Internet of Things (IoT), artificial intelligence (AI), digital twins, industrial robots and additive manufacturing to facilitate smart factories, while redefining technological progress as a mean to empower human workers and reduce environmental impact [2]. The focus of I 4.0 was predominantly on efficiency and connectivity, which gave rise to concerns regarding job security, rising unemployment due to automation, and environmental issues such as excessive energy consumption and electronic waste [1]. I 5.0 therefore aims to address these social and ecological challenges by leveraging human capabilities through collaborative machine systems, as opposed to replacing them [2].

Collaborative robots (Cobot) are designed to undertake repetitive or hazardous tasks in contact-rich environments, thereby allowing human operators to focus on more complex activities such as oversight, problem solving or in general operations that add greater value [1]. This shift presents a range of practical opportunities, including mass customization, greater production flexibility, optimized resource use, and the inclusion of disabled people in the workforce. These opportunities can further help to meet growing consumer demand for personalized products while lowering material and energy footprints [2]. Simultaneously, the increased proximity of humans and machines gives rise to new and significant safety challenges. These challenges require robust, easy-to-handle technical solutions as well as regulatory guidance to ensure worker protection under all conditions [4]. In practice, common safety systems mostly rely on compliance control and collision detection [5] such as torque sensors in order to provide reliable collision detection and force control. However, they only signal after an impact has occurred. Complementary approaches such as proximity sensing offer promising opportunities for achieving predictive protection and facilitating smoother and safer human-robot interaction (HRI) [4].

From an occupational health and safety perspective, human-robot collaboration (HRC) has been shown to reduce (musculoskeletal) health risk factors, decrease physical effort, improve coordination and efficiency as well as lower exposure to hazards [4]. The study from Fournier et al. [4] also reports fewer errors per unit of time and maintained trust in cobotic systems, even when total error counts remain similar across setups.

By effectively lowering physical barriers to enter the primary labor market, these ergonomic benefits lay the foundation for a more inclusive workforce. Beyond the individual health benefits, the ability to serve diverse physical impairments fosters compliance with legal frameworks. For example, § 154 of the German Social Code Book IX [6] promotes inclusion and reasonable accommodations for severely disabled employees. To ensure that the previously mentioned objectives of I 5.0

Corresponding author: S. C. Keunecke (keunecke@igmr.rwth-aachen.de)

are aligned with the general goals of social participation and equity, the implementation of Cobots must be realized in a manner that is both safe and anticipatory. A critical and first step for ensuring safety by collision avoidance in HRC is the ability to reliably detect objects real-time in proximity.

II. Related Work

The framework for HRC as outlined in ISO TS 15066 categorizes four distinct interaction methods. However, no current robotic system has been designed to implement all four modes simultaneously [11]. Figure 1 illustrates the scope of the challenges that a Cobot must overcome in order to meet those requirements. To address these varied needs of ISO TS 15066, object and collision detection systems play a fundamental role. As seen in Figure 2, sensing modalities in Cobots are either classified as proprioceptive or exteroceptive [7] and as intrinsic or extrinsic. While exteroceptive sensing is essential for monitoring the Cobot workspace, it can be further divided into extrinsic and intrinsic perspectives [13]. Extrinsic configurations offer a comprehensive environmental context. Yet, these sensors encounter substantial deployment challenges. They exhibit heightened sensitivity to changes in placement, necessitate ongoing calibration, and are vulnerable to occlusion in complex factory environments [7], [13]. Additionally, substantial data fusion and volume introduces processing latency that can degrade system response and effectiveness in real-time deployment [7]. In contrast, the utilization of intrinsic configuration has been proposed as a method to address these concerns. The simulation study by Kumar et al. [10] confirms that by minimizing data volume and deployment complexity, intrinsic setups significantly improve system response time, leading to measurable enhancements in safety and productivity compared to traditional static collision avoidance systems. However, research on 3-D sensing devices mounted directly on the robot is limited beyond 2-D scanning LiDARs [7].

In the context of 3-D perception, industry applications often utilize structured light or ToF sensors [13]. ToF sensors are particularly attractive for onboard deployment due to their compact hardware, low processing requirements, and high refresh rates [13]. Unlike standard visual sensors, ToF devices exhibit minimal sensitivity to variations in image intensity, thereby providing immediate depth information, even for objects in motion [7], [8], [17]. These characteristics have led to their use in HRC for point cloud generation, pose tracking, and affordance estimation [13]. However, ToF sensors encounter limitations, especially in outdoor settings, due to factors such as sunlight, sensitivity to reflective materials, and potential interference from neighboring devices [13].

Another significant challenge in utilizing depth sensors lies in the representation of data. Raw point clouds enable detailed geometric descriptions. However, they are inherently unorganized, potentially noisy due to sensor physics, and exhibit variable density based on range [15]. While denoising techniques, including Moving Least Squares and Geometric Deep Learning, have been developed, these methods are computationally intensive and or require substantial training data sets, leading to bottlenecks in real-time monitoring and dynamic replanning [15]. In order to address these scalability limits, voxel representation is commonly adopted [15]. Voxel grids provide a structured format that simplifies occupancy queries and enables faster obstacle detection by trading spatial detail for processing efficiency [15]. Although voxelization into fixed grids is widely used due to its simplicity, there is a lack of robust methods for converting these 3-D maps into occupancy representations [14].

To manage spatial and temporal data within these grids, clustering algorithms such as DBSCAN and k-means can be explored. However, existing research indicates that these methods can encounter challenges in preserving feature continuity and local object details when applied to complex point clouds or voxel grids [16]. Consequently, the specific potential of applying DBSCAN directly to super sparse voxel structures in a low-cost setup to retain the necessary feature continuity while meeting the computational efficiency requirements of real-time HRC remains underexplored.

Therefore, we propose a low latency motion tracking system for HRC by combining onboard ToF sensing in a zero shot approach with a super sparse voxel clustering pipeline utilizing DBSCAN.

III. Methodology

A. Hardware Setup

The deployment of sensors within complex, dense environments, such as production lines, requires a low-cost and compact form factor sensor to ensure the robot's freedom of movement. In addition, the application demands high measurement accuracy, a fast sampling rate, and robustness against external interference. Based on these requirements and the sensor evaluation study by Keunecke et al. [18], the decision was made in favor of the ToF sensor.

Figure 3 provides an overview of the hardware configuration, with particular emphasis on the custom sensor ring. The custom 3D-printed ring module integrates seven VL53L7CX Time-of-Flight (ToF) sensors, with each sensor featuring an 8×8 grid. These sensors are connected to a single Raspberry Pi Pico (RP2040) microcontroller, which transmits the collected data through a single Ethernet connection to the laptop.

Regarding placement, the ring is mounted at the center of the second link of a Universal Robots UR10e. This configuration is designed to ensure an unobstructed field of view (FOV) of the region of interest, which is $1500 \times 1400 \times 1600\ \mathrm{mm}^3$ big. It also minimize self-occlusion in the static, upward-facing configuration depicted in Figure 3, meaning preventing the sensors from detecting the robot arm itself.

B. Data Processing Pipeline

As illustrated in Figure 4, the proposed motion detection and tracking pipeline transforms sensor data into a structured temporal voxel representation, allowing real-time object detection. Although raw point clouds offer high spatial precision, a discrete voxel-based approach is implemented to significantly

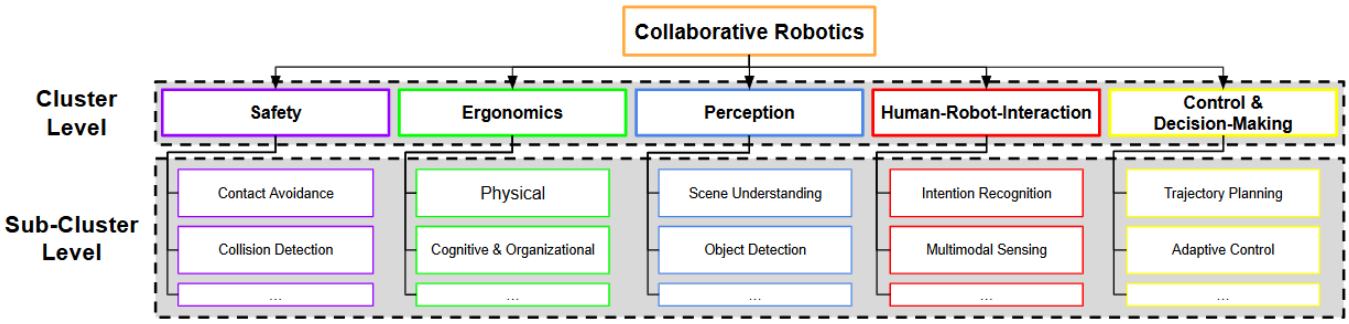

Fig. 1: Classification of Core Areas and Subdomains in Collaborative Robotics

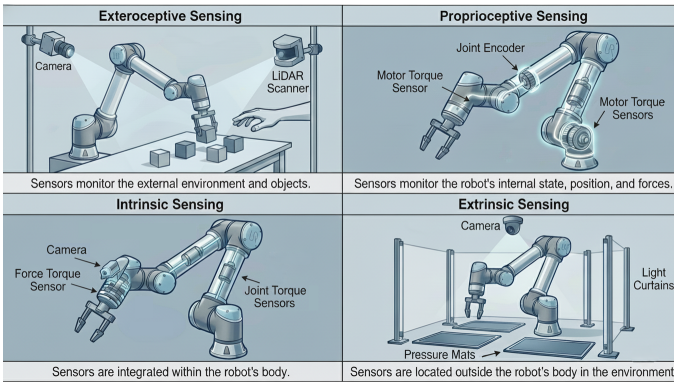

Fig. 2: Sensing Modalities in Collaborative Robotics

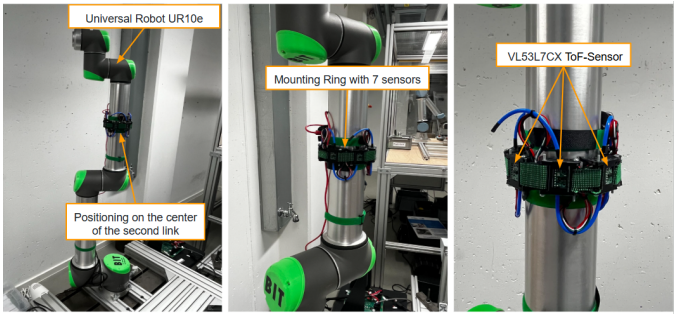

Fig. 3: Setup Overview of the Cobot

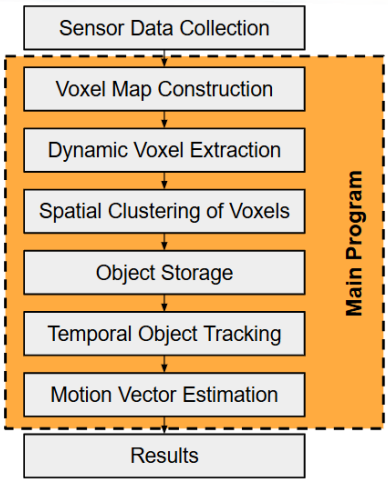

Fig. 4: Architecture of the Proposed Voxel Tracking and Motion Pipeline

improve computational performance as it is critical for low-latency HRI. The pipeline begins with inputting sensor data in JSON-format. This data is then mapped into a global coordinate system. During voxelization, the real-world coordinates are rounded to the nearest grid index based on a fixed resolution. This process creates a 3-D occupancy grid that enables rapid spatial analysis.

Afterwards, the system uses a temporal filtering mechanism to differentiate between static surroundings and moving obstacles (see section III-C). Each voxel in the grid stores the timestamp of its most recent activation. The pipeline extracts only dynamic voxels by comparing these timestamps against a defined $TimeTolerance$ and then passes them to the spatial clustering stage. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is employed for this purpose. In this implementation, the DBSCAN hyperparameters, specifically the neighborhood radius ϵ and the minimum number of points $MinPts$ required for a core point, are optimized based on the discrete grid topology (see section III-D). This ensures that structures are reliably detected as coherent objects.

The detected spatial clusters are stored as individual object instances. Instead of simplifying an object to a single centroid at this stage, the system retains the full set of associated voxels. This is critical for preserving the spatial dimensions of the obstacle, ensuring that complex geometries remain distinguishable, and providing more features for subsequent tracking. Temporal tracking is implemented through multi-stage association logic. The primary method relies on spatial overlap (see section III-E). The system checks whether the voxels of a newly detected cluster intersect with the recent history of an existing tracked object. For high-velocity obstacles, where spatial overlap between frames may not occur. Another method that is used is a nearest-neighbor search within a defined radius.

Once an object is identified in consecutive time frames, the pipeline calculates its motion vector. This calculation is

derived from the displacement of the object's geometric center relative to the elapsed cycle time. To ensure robustness against jitter, the estimated velocity is averaged over the most recent frames. These motion vectors provide the current velocity state and serve as the basis for rudimentary trajectory prediction. Finally, the pipeline outputs the determined movement vectors and the approximate spatial bounds of related objects. This provides the necessary data to perform proactive collision avoidance in dynamic environments.

C. Preprocessing of Static Environment

Reliable dynamic object detection in voxel-based environments depends on the quality of the input data. Preliminary experiments revealed that applying DBSCAN directly to raw voxel data produced inconsistent outputs. A primary source of error are persistent background voxels which create two distinct artifact types: temporal ghosting and spatial merging. As demonstrated in Figure 5, static voxels remain in the environment scan, causing the formation of ghost objects that do not correspond to physical objects. More importantly, these persistent points often act as spatial connectors between different moving objects or between a moving object and the background when clustering is applied. This effect causes distinct objects to merge into a single cluster, preventing accurate tracking of the objects individually.

To address the spatial and temporal artifacts, a temporal occupancy filter has been implemented. Each voxel V stores an integer Persistence Score $P_t(V)$, which is updated once per cycle, and a most recent activation timestamp $T(V)$. Following parameters were chosen for the experiments: Increment Factor $\alpha = 2$, Decay Factor $\beta = 1$, Persistence Threshold $\tau = 10$ cycles and Time Tolerance $\Delta T = 0.08$ s.

When voxel V is observed at time t the persistence score $P_t(V)$ increases the stored score by α only if the current score is strictly below τ:

$$P_t(V) = \begin{cases} P_{t-1}(V) + \alpha & \text{if } V \text{ observed at } t \ \wedge \ P_{t-1}(V) < \tau, \\ P_{t-1}(V) & \text{otherwise} \end{cases}$$

The classification of active voxels is determined by the score $P_t(V)$. It combines the current sensor input via the α-increment and serves as the decision score for the current cycle t. This representation captures the immediate, unprocessed state of the voxel following the completion of the sensor reading process. Following this step, a decay factor β is applied to recently active voxels. This is done to prevent temporal ghosting.

The decay is applied if $\Delta t(V) < \Delta T$, where $\Delta t(V) = t_{\text{now}} - T(V)$. The post-decay score, $P_t^*(V)$, is defined as the successor state utilized for the subsequent cycle $(t+1)$. This score is critical as it enforces the temporal hysteresis and ensures that temporary voxels are gradually eliminated.

$$P_t^*(V) = \begin{cases} \max(P_t(V) - \beta, 0) & P_t(V) > 0 \ \wedge \ \Delta t(V) < \Delta T, \\ P_t(V) & \text{otherwise} \end{cases}$$

The classification logic below determines the voxel's state based on two criteria: Score persistence and temporal recency. The implementation classifies a voxel as static based on the decision score $P_t(V)$ when:

$$\text{State}(V) = \begin{cases} \text{Static} & \text{if } P_t(V) \geq \tau \ \vee \ \Delta t(V) \geq \Delta T, \\ \text{Dynamic} & \text{if } P_t(V) < \tau \ \wedge \ \Delta t(V) < \Delta T. \end{cases}$$

This classification operates on the undecayed score $P_t(V)$, prior to the state update that calculates $P_t^*(V)$. The sequential structure ensures that the classification decision, which immediately determines the input for the clustering phase, is based on the most accurate and up-to-date sensor cycle. Moreover, the conservative static filtering ensures that the static decision is based on the peak persistence score $P_t(V)$ achieved in the current cycle, resulting in a conservative filter. A voxel is flagged as static the instant it hits the threshold, preventing it from influencing the clustering results, even if the subsequent decay would immediately pull the successor score below the threshold. This prioritization minimizes the risk of spatial merging artifacts by aggressively removing potential environmental connectors. Furthermore, to prevent long-term memory pollution and ensure stability, the memory resets all scores below the threshold to zero every 5 s.

D. DBSCAN Hyper Parameter Tuning with k-distance Plot

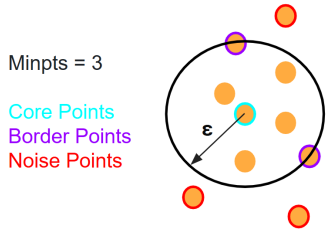

Fig. 6: Schematic Overview of DBSCAN Logic

To cluster the voxel data and filter outliers, we employ the density-based clustering algorithm DBSCAN [12] due to its combination of time complexity in the given data structure and algorithmic simplicity. As shown in Figure 6, DBSCAN classifies data points into three classes: core, border, and noise points based on the local point-density. Defined by a search radius ϵ and a density threshold $MinPts$, a cluster is formed by recursively connecting density-reachable points. This approach allows for the detection of arbitrarily shaped clusters while effectively isolating sparse data points as noise [12]. In order to determine the optimal hyperparameters ϵ and $MinPts$ for the DBSCAN based on a sample of different scenarios next to the Cobot, the following approach is implemented:

The DBSCAN hyperparameters were selected using k-distance plot analysis. As shown in Figure 7, the graph displays a distinct staircase pattern because the voxelized point clouds are defined on a discrete integer grid. In this metric space, the Euclidean distance d between the centers of two voxels is quantized and computed as $d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$, where $\Delta x, \Delta y, \Delta z$ are integer

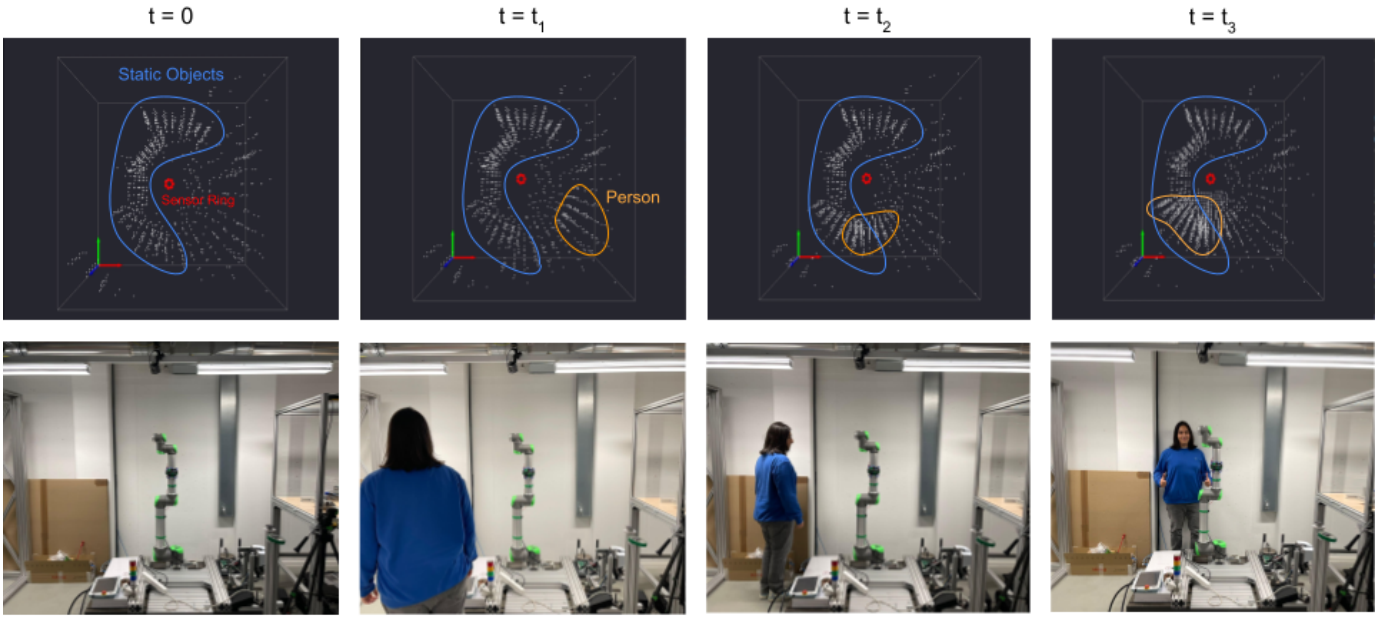

Fig. 5: Bird's Eye View of Dynamic Object Ghosting in Voxel Map

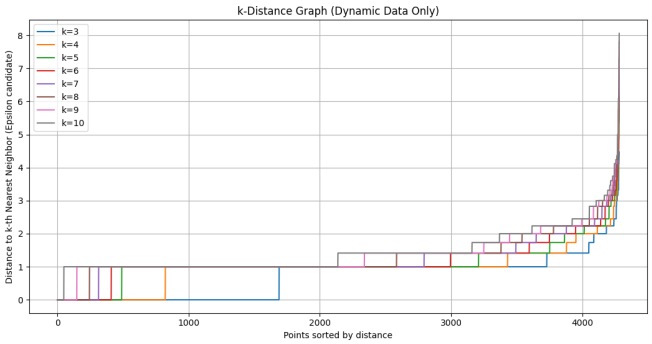

Fig. 7: k-distance Plot for Multiple k-values

differences in Voxel Units. Consequently, the possible distance values are discrete and limited. Therefore, many points share identical k-distance values creating extended horizontal segments. Figure 7 visually confirms the effect of these discrete distance values, where the curves jump between stable plateaus. This artifact shows that small variation of ϵ within a flat segment does not affect the neighborhood assignments at all. However, choosing the value of ϵ across a jump point can abruptly and simultaneously alter the number of neighbors for many voxels, which can lead to significant changes in the cluster structure.

Difference $(\Delta x, \Delta y, \Delta z)$	Distance d	Description
$(1, 0, 0)$	$\sqrt{1}$	Face-to-face Neighbor
$(1, 1, 0)$	$\sqrt{2}$	2-D Diagonal
$(1, 1, 1)$	$\sqrt{3}$	3-D Diagonal
$(2, 0, 0)$	$\sqrt{4}$	Two-Voxel Separation

TABLE I: Euclidean Distances between Voxel Centers

To justify the choice of ϵ, we analyze the discrete distance values available on the integer grid, as summarized in Table I. The distance value $d = \sqrt{3} \approx 1.73$ represents the longest Euclidean distance between the center of a voxel and any of its immediate neighbors, specifically the 3-D diagonal corner neighbor (cf. $(1, 1, 1)$ in Table I). This analysis shows a topological gap in the Voxel Unit space between $d \approx \sqrt{3} \approx 1.73$ and the next possible discrete distance value, which is $d = 2.0$. If ϵ were chosen as $\sqrt{3}$, DBSCAN would strictly connect only the densest local neighbors. Thus, selecting $\epsilon = \sqrt{4}$ is a reasonable choice, as this value precisely includes the distance value of $\sqrt{4}$, thereby enabling two-hop connectivity. This configuration allows the algorithm to bridge small holes and discretization artifacts by linking structures separated by one empty voxel without merging physically distinct obstacles. The vertical coordinate at the elbow is typically chosen as ϵ for the corresponding $MinPts$ value in DBSCAN [12], [16]. However, selecting ϵ at a jump point can produce abrupt differences in clustering, therefore it is recommended to select ϵ near the top of a stable flat segment immediately before a jump, or to use automated elbow detection methods based on curvature rather than manual inspection [16]. The k-distance curves in Figure 7 demonstrate a clear elbow point near the value of $\epsilon = 2.0$. This experimental validation strongly supports the theoretical reasoning for $\epsilon = 2.0$.

Despite these discretization effects, our analysis further shows structural stability for $k \in [3, 10]$. In addition Figure 7 shows, the curves for $k \geq 5$ follow nearly identical trajectories and exhibit consistent elbow locations, representing the transition from dense cluster interiors to sparse noise [12]. Increasing $MinPts$ trades sensitivity for noise suppression, where larger values filter transient artifacts at the cost of increasing the

minimum physical size required for a valid cluster. Based on the convergence of the curves, consistent elbow point across the evaluated k-range, and the need to eliminate sparse outliers, we selected $MinPts = 8$ as the robust setting.
These parameters, ϵ = 2.0 and $MinPts = 8$, were verified empirically for given sample of close scenarios to ensure robustness for the specific Cobot and sensor configuration used in this study. It is important to emphasize that hyperparameter optimization is highly sensitive to sensor quality, data density, and the spatial characteristics of sample scenarios. Therefore, the final parameter selection remains task-specific and aligned with the operational goals and environmental constraints of the Cobot application.

E. Temporal Object Tracking and Velocity Constraints

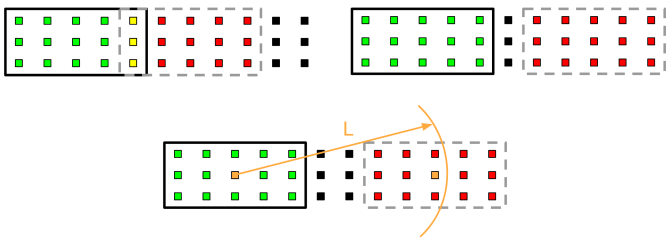

Fig. 8: Overlapping Technique of Tracking Objects

- Gray Boxes: Object Position at $t = 0$
- Black Boxes: Object Position at $t = 1$
- Black Dots: Inactive voxels
- Red Dots: Active Voxels at $t = 0$
- Green dots: Active Voxels at $t = 1$
- Yellow Dots: Active Voxels at $t = 0$ and $t = 1$
- Orange Dots: Center of Area of Object

1) Tracking of Objects in Voxel Map: The temporal object tracking mechanism identifies and tracks dynamic objects within the workspace, fundamentally relying on the spatio-temporal coherence of active voxels. The core of the object tracking utilizes an overlap-based associative tracking methodology, as illustrated in Figure 8. An object cluster detected at time t is associated with a previously tracked object from time $(t-1)$ if a sufficient spatial overlap exists between their respective voxel sets. This condition is checked over the recent voxel history within a tolerance $\Delta T = 80\,\text{ms}$. This approach is computationally efficient, avoiding complex distance computations in favor of set intersection checks. If $V(t)$ is the voxel set of a new cluster and $V_{obj}(t_{recent})$ is the accumulated set of a known object's recent voxels, the object is maintained if $|V(t) \cap V_{obj}(t_{recent})| > 0$.
A constraint to this method is the maximum detectable velocity v_{max}. The velocity v of the object must be constrained such that it lies below the threshold of the ratio of the object's exposed length, defined as L with respect to the direction of motion and the designated tracking cycle time t. If the velocity exceeds this threshold, $v > L/\Delta t$, the object cluster at time t will have shifted spatially such that it no longer overlaps with its past position at $(t-1)$. In this scenario, the system incorrectly interprets the moving object as two separate, losing the temporal association and setting the estimated velocity to near zero.
To avoid the limitations of pure overlap tracking, a proximity check is executed for clusters that fail the initial method. This secondary method attempts association based on the distance between the cluster centroid at time t and the centroid of a known object at $(t-1)$.This approach helps bridge small temporal gaps, in which an object's movement is marginally faster than the overlap threshold, yet remains within a reasonable proximity range. The current implementation utilizes a maximum distance of two voxels.

2) Velocity estimation: Velocity estimation is performed by calculating the vector difference between the object's centroid at the latest two time steps, $\mathbf{v} \approx (\mathbf{C}_t - \mathbf{C}_{t-1})/\Delta t$, where $\mathbf{C}$ are the real-world coordinates of the centroids. This displacement is then compared against a predefined velocity threshold, $v_{thres} = 20.0\,\text{mm/s}$, to differentiate genuine motion from sensor noise and residual static jitter.

IV. Experimental Results

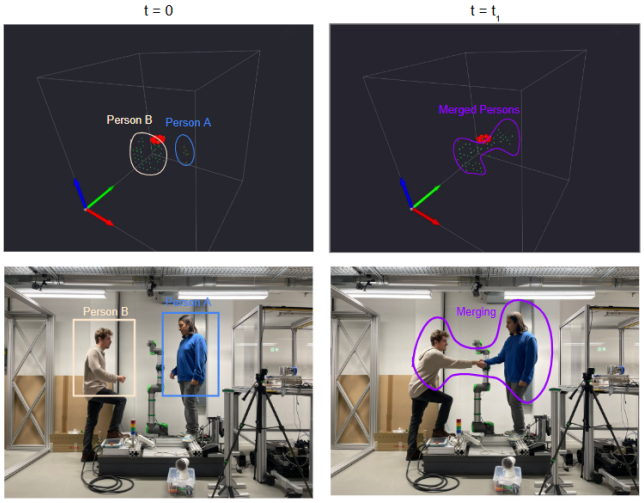

Fig. 9: Proximity Induced Object Merging in DBSCAN

During the evaluation, the pipeline's cycle time emerged as a primary constraint, prompting a comparison between conventional Boolean-based voxelization and a continuous, time-based approach. In theory, the Boolean method improves computational efficiency by replacing complex floating-point arithmetic with low-latency bitwise operations. Although this advantage has yet to be fully validated empirically in specific hardware environments, the Boolean approach provides a robust framework for movement reconstruction by treating each time frame as an independent, discrete state. Unlike time-surface methods, which often overwrite historical data with the most recent timestamps, our discrete separation method preserves the complete occupancy history. This enables significantly more granular and extended retrospective

trajectory analysis. Overall, the time-based method offers more consistent tracking, greater adjustability via ΔT,, and better scalability for multi-sensor configurations because it can fluidly memorizes incoming data.

After initial processing bottlenecks were resolved, motion vector estimation showed a high correlation with ground-truth movements. However, significant noise interference occasionally resulted in erratic findings where the estimated direction was accurate, but the velocity magnitude showed high variance. The tracking system, which is operated by a Raspberry Pi Pico acquires raw data at a frequency of $8.33\,\text{Hz}$, achieves an internal processing frequency of 14 to $16\,\text{Hz}$ (with a cycle time of approximately $65\,\text{ms}$). This performance is critically limited by serial transfer rates and loop execution time. As established in the theoretical framework, the maximum velocity v_{max} that can be reliably tracked, is a function of the object's geometry and cycle time. Velocities exceeding 1,300 mm/s were observed to cause significant overlapping artifacts, leading to erratic tracking behavior. Furthermore to maintain signal integrity in the lower spectrum, a velocity threshold of $v_{thres} = 20.0\,\text{mm/s}$ was implemented to filter out low-speed noise.

The pipeline's key achievement is the demonstration of a low-cost, real-time, low-latency tracking system that operates within a super-sparse voxel environment in a zero-shot approach. As shown in Figure 9, the system can simultaneously detect and track two people performing a handshake in close proximity to the Cobot. This result showcases the pipeline's ability to separate and track distinct objects. This confirms its potential for collision avoidance, despite it's limitations.

A. Limitations

Despite the use of overlapping mechanisms and dynamic filtering, the system exhibits erratic behavior in scenarios. This is primarily due to the sensitivity of the DBSCAN clustering algorithm. In data-scarce environments, sensor noise and false positives significantly distort cluster formation. Since motion vectors are calculated based on the geometric centroids of these clusters, errors in the clustering process propagate directly to velocity estimation, resulting in "non-physical" movement vectors. Additionally, the system's detection efficacy depends heavily on the target's profile, including its distance, relative velocity, and material properties (e.g., signal absorption in translucent surfaces) [9]. Empirical data from our experiments show a steep decline in reliability for low-profile geometries with $d < 2\,\text{cm}$. While dynamic targets benefit from increased spatial sampling across multiple sensor FOV, establishing a precise, velocity-based detection threshold is difficult due to the nonlinear correlation between spatial precision and signal quality across varying ranges.

Finally, the current hyperparameter configuration, specifically the ϵ and $minPts$ values for DBSCAN, is the result of local tuning for our specific sensor configurations and task environment. Without a global optimization framework, the system may require manual recalibration when deployed in significantly different spatial contexts or with alternative sensor geometries, data rate, etc..

B. Outlook

Future work will focus on applying morphological dilation to cluster volumes prior to checking for spatial overlap. This approach promises to be more computationally efficient for proximity-based association. Additionally, we aim to expand the pipeline by incorporating additional sensor rings to provide richer data density and reduce issues associated with sparse voxel fields, thereby boosting the pipeline's performance.

Lastly, the implementation of an automated, distance-aware parameter optimization scheme is planned. By dynamically adjusting clustering parameters based on the observed noise and distance to the target, it is expected that the need for manual tuning will be eliminated and the robustness of the motion vector estimation in diverse dynamic environments as well feature-rich object detection will be enhanced.

V. Conclusion

In this paper, a novel, low-cost intrinsic sensing system is presented. This system is designed to address the limitations of reactive force and predictive collision avoidance in Industry 5.0 environments. By integrating a multi-sensor ToF array with a high-performance temporal voxel pipeline, the potential for real-time object tracking within super-sparse data structures has been demonstrated. The methodology applied for temporal occupancy filtering effectively resolves the chronic issues of ghosting and spatial merging that typically plague density-based clustering in discretized spaces. Moreover, the integration of a topologically-grounded hyperparameter selection for DBSCAN offers a robust theoretical foundation for the development of future low-latency HRC applications.

While the system demonstrated a high degree of correlation with ground-truth movements and displayed robustness in close-proximity interactions, critical limitations were identified with regard to sensitivity and low-profile geometric detection ($d < 2$ cm). Consequently, future work will concentrate on integrating adaptive background subtraction to accommodate semi-static environments and developing a global optimization framework for hyperparameters to ensure cross-platform scalability. This research offers a foundation for cobotic workspaces that are safer, more resilient, and inclusive. It aligns technological advancement with the human-centric priorities of the primary labor market.

VI. Acknowledgment

This work was conducted within the project IIDEA, Inclusion and Integration through Cobots in the Primary Labor Market. Additional information on the project can be found on the official IIDEA project page: IIDEA Website.

This research was funded by the German Federal Ministry of Labor and Social Affairs through resources from the Compensation Fund.

The authors acknowledge the use of Gemini Nano Banana Pro to assist in the visual rendering of Figure 2 based on specific structural instructions provided by the authors.

REFERENCES

[1] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "From automation to collaboration: exploring the impact of industry 5.0 on sustainable manufacturing", *Discover Sustainability*, vol. 6, no. 1, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s43621-025-01201-0

[2] M. Breque, L. De Nul, and A. Petridis, "Industry 5.0: Towards a sustainable, human-centric and resilient European industry", European Commission, Directorate-General for Research and Innovation, Luxembourg, Policy Brief, 2021. [Online]. Available: https://data.europa.eu/doi/10.2777/308407

[3] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Towards Industry 5.0: A Survey on Human-Robot Collaboration", *Actuators*, vol. 9, no. 6, p. 113, 2020. [Online]. Available: https://www.mdpi.com/2075-1702/9/6/113

[4] É. Fournier, C. Jeoffrion, B. Hmedan, D. Pellier, H. Fiorino, and A. Landry, "Human-Cobot collaboration's impact on success, time completion, errors, workload, gestures and acceptability during an assembly task," *Applied Ergonomics*, vol. 119, p. 104306, Sep. 2024 [Online]. Available: https://doi.org/10.1016/j.apergo.2024.104306

[5] J. Lee, Y. Kim, S. Kim, Q. Nguyen and Y. Heo, "Learning Fast, Tool-Aware Collision Avoidance for Collaborative Robots," *arXiv preprint arXiv:2508.20457*, 2025. [Online]. Available: https://arxiv.org/pdf/2508.20457

[6] Bundesministerium der Justiz, "Sozialgesetzbuch (SGB), Neuntes Buch (IX) – Rehabilitation und Teilhabe von Menschen mit Behinderungen, §154 Pflicht der Arbeitgeber zur Beschäftigung schwerbehinderter Menschen", 2018. [Online]. Available: https://www.gesetze-im-internet.de/sgb_9_2018/__154.html

[7] O. A. Adamides, A. S. Modur, S. Kumar und F. Sahin, "A Time-of-Flight On-Robot Proximity Sensing System to Achieve Human Detection for Collaborative Robots", in *Proceedings of the IEEE 15th International Conference on Automation Science and Engineering (CASE)*, Vancouver, BC, Kanada, 2019, pp. 1230–1236.

[8] R. Ahmad und P. Plapper, "Human-Robot Collaboration: Twofold Strategy Algorithm to Avoid Collisions Using ToF Sensor," *International Journal of Materials, Mechanics and Manufacturing*, vol. 4, no. 2, pp. 87–92, May 2016. [Online]. Available: https://www.ijmmm.org/vol4/243-MA030.pdf

[9] C. Sifferman, M. Gupta and M. Gleicher, "Efficient Detection of Objects Near a Robot Manipulator via Miniature Time-of-Flight Sensors," *arXiv preprint arXiv:2509.16122*, 2025. [Online]. Available: https://arxiv.org/pdf/2509.16122

[10] S. Kumar, C. Savur, and F. Sahin, "Dynamic awareness of an industrial robotic arm using time-of-flight laser-ranging sensors," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 2850–2857. doi: 10.1109/SMC.2018.00485

[11] U. B. Himmelsbach, T. M. Wendt and M. Lai, "Towards Safe Speed and Separation Monitoring in Human-Robot Collaboration with 3-D Time-of-Flight Cameras," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, CA, USA, 2018, pp. 197–200. doi: 10.1109/IRC.2018.00042

[12] M. Ester, H. P. Kriegel, J. Sander und X. Xu, "A density based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD)*, Portland, OR, USA, 1996, pp. 226–231.

[13] S. Kumar, C. Savur and F. Sahin, "Survey of Human–Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 280–297, Jan. 2021. doi: 10.1109/TSMC.2020.3041231

[14] S. Fredriksson, A. Saradagi und G. Nikolakopoulos, "Voxel map to occupancy map conversion using free space projection for efficient map representation for aerial and ground robots," *arXiv preprint arXiv:2406.07270*, 2024. [Online]. Available: https://arxiv.org/abs/2406.07270

[15] J. Chlebek et al., "Optimized Grid Voxelization for Obstacle Avoidance in Collaborative Robotics," *IEEE Access*, vol. 13, pp. 45187–45197, 2025. doi: 10.1109/ACCESS.2025.3549622

[16] H. Liu, Y. Tang, and H. Wang, "Robust and Fast Point Cloud Registration for Robot Localization Based on DBSCAN Clustering and Adaptive Segmentation," *Sensors*, vol. 24, no. 24, p. 7889, 2024. doi: 10.3390/s24247889

[17] S. Natarajan, A. Vogt and F. Kirchner, "Dynamic Collision Avoidance for an Anthropomorphic Manipulator Using a 3-D ToF Camera," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, Munich, Germany, 2010, pp. 1–7.

[18] S. C. Keunecke, M. Hüsing, and B. Corves, "Sensorauswahl für die sichere Arbeitsraumüberwachung von Robotern durch integrierte Umfelderfassung Sensor Selection for Safe Workspace Monitoring of Robots through Integrated Environmental Sensing," in *11. IFToMM D A CH Konferenz 2025, 20./21. Februar 2025, FH Kärnten, Villach*, Villach, Austria, 2025. doi: 10.17185/duepublico/82920