Reactive Collisions Avoidance in Collaborative Robot Environment

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*—XYXYX**

The code is available at https://github.com/philippluedtke/predictive_collision_avoidance.

***Index Terms*—ToF Sensors; Voxels; Collision Avoidance; Human-Robot Collaboration; Real-Time Robot Control; Hyper Parameter Optimization; Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**

I. Introduction

Industry 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 value [1]. Building on the digital and cyber-physical foundations of previous industrial advancements, especially Industry 4.0, Industry 5.0 utilizes technologies such as the industrial Internet of Things (IoT), artificial intelligence (AI), digital twins, industrial robots and additive manufacturing to facilitate smart factories, while redefining technological progress as a means to empower human workers and reduce environmental impact [2]. The focus of Industry 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]. Industry 5.0 aims to address these social and ecological gaps by augmenting 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 and more value-adding operations [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 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 and regulatory guidance to ensure worker protection under all conditions [4]. In practice, common safety systems rely on compliance control and collision detection [6] such as torque sensors in order to provide reliable collision detection and force control. However, they only signal after an impact has occurred. This means that complementary approaches such as proximity sensing offer promising opportunities for achieving anticipatory protection and facilitating smoother and safer human-robot interaction [4]. From an occupational health and safety perspective, human-robot collaboration (HRC) has been shown to reduce (musculoskeletal) risk factors, decrease physical effort, improve coordination as well as efficiency, and lower exposure to hazards. An experimental study also reports fewer errors per unit of time and maintained trust in cobotic systems, even when total error counts remain similar across conditions [4]. In addition, the societal relevance of human-centered manufacturing is reinforced by legal frameworks such as the § 154 of the German Social Code Book IX that promote inclusion and reasonable accommodations for severely disabled employees. This ensures that industrial practice aligns with broader goals of social participation and equity [5].
In order to achieve the above-mentioned goals of Industry 5.0 and to meet the policies that include the inclusion of people with disabilities in the workforce, it is necessary to implement CoBots in a safe and anticipatory manner. A critical and first component for ensuring collision avoidance in HRC is the ability to detect objects in proximity [9].

Therefore, we propose a real-time motion detection system for HRC using DBSCAN in super sparse voxel occupancy environment utilizing on-board Time-of-Flight (ToF) sensors. This system allows us to detect moving and static objects in proximity to the robot arm using a combination of the density-based clustering algorithm DBSCAN and object tracking using overlapping techniques.

II. Related Work

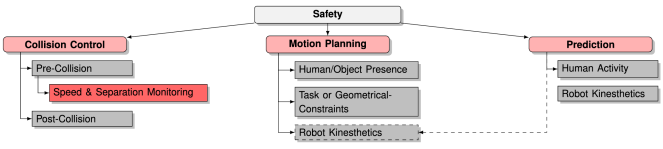


Fig. 1: Tree of Safety Approaches in HRC

The regulatory 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

This research has been funded by ...
Contact: ...

four modes simultaneously [11]. To address these varied needs, obstacle and collision avoidance systems are classified into proprioceptive and exteroceptive architectures [7]. While exteroceptive sensing is essential for monitoring the workspace, it is further divided into extrinsic and intrinsic perspectives [13]. Extrinsic sensors, including 3-D LiDARs and motion capture systems, 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, the substantial data volume generated by external networks imposes processing overheads that degrade system response and effectiveness in real time deployment [7]. In contrast, the utilization of intrinsic range sensing has been proposed as a method to address these concerns. The findings of the simulation study by Kumar et al. suggest that intrinsic setups result in quantifiable enhancements in safety and productivity when compared to static collision avoidance systems [10]. 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 frequently 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]. Contrary to 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 in outdoor settings due to factors such as sunlight, sensitivity to reflective materials, and potential interference from concurrent devices [13].

A significant challenge in utilizing depth sensors lies in the representation of data. Raw point clouds enable detailed geometric descriptions. However, they are inherently unorganized, 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, 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 data within these grids, clustering algorithms such as DBSCAN and k-means have been explored as alternatives to constant grid sizes [16]. However, existing research indicates that these methods frequently encounter challenges in preserving feature continuity and local object details when applied to complex point clouds [16]. Consequently, the specific potential of applying DBSCAN directly to pre-filtered, super sparse voxel structures to retain feature continuity while meeting the computational efficiency requirements of real-time HRC remains underexplored [16]. This work addresses this gap by combining onboard ToF sensing with a sparse voxel clustering pipeline utilizing DBSCAN.

III. Methodology

A. Systems Hardware

The proposed object detection algorithm was tested on a sensorized ring module composed of seven ToF-based sensing units, designed to be mounted at the center of the second link of a robotic arm. The overall system architecture of the robot consists of the following components:

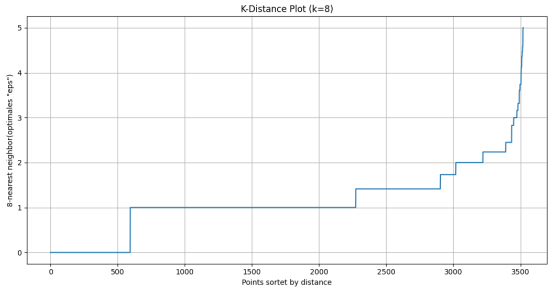
- Microcontroller: Raspberry Pi Pico (RP2040)
- Sensor: Time of Flight Senesor VL53L7CX ToF (8x8 grid)
- Mounting: One ring made out of six sensors (with dead zones)
- Robotic Arm U10

B. Preprocessing of Static Objects

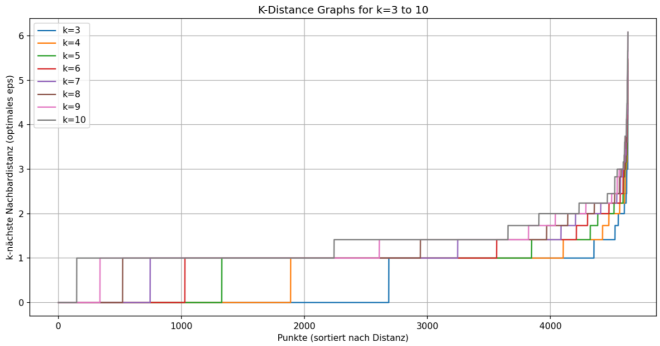
Testing with real-time data input led to the identification of various implementation as well as preliminary design flaws in our approach. Despite the program's effective management of the increased data input from six additional sensors, with cycle times ranging from 65 to 75 milliseconds, significant concerns remained. The ability to consistently detect and distinctly differentiate between various objects was found to be inconsistent. A significant cause of these errors was the static surroundings' persistent tendency to group together with nearby moving objects or to merge and subsequently unmerge with other static objects. This behavior led to the distortion of movement and resulting clustering errors as seen in Figure X. However, a substantial adjustment was implemented through the integration of a preprocessing step. Prior to executing the main program, a scan of the environment for static objects is performed. Subsequently, it is assumed that these static voxels do not contain relevant information for object detection. This approach led to a substantial reduction in merging errors as well as noises. However, the method for identifying static objects remains underexplored and requires further development such as a continuous background scan.

C. Tunning Hyper Parameters of DBSCAN

In order to determine the optimal hyperparameters for the DBSCAN algorithm (eps and min samples), we initially employed analytical optimization techniques. The K distance plot is a widely used diagnostic tool for selecting the epsilon parameter for density-based clustering algorithms including DBSCAN. As seen in Figure X and Y, the plot not only conveys a scale for neighborhood density for voxel-based point clouds but also signatures of the underlying grid structure. Mark X of Figure X showcases the elbow point, which is the location of maximal curvature in the K distance curve.



(a) k Distance Plot with single value k = 8



(b) k Distance Plot with multiple values of k

Fig. 2: k Distance Plot from limited random sample

In practice, it is the point where the plotted distances change from a relatively flat trend to a markedly increasing slope. Geometrically speaking, this point separates voxels that reside in dense local neighborhoods from points that are isolated or belong to sparse clusters []. Therefore, the vertical coordinate at the elbow is commonly chosen as the epsilon value for DBSCAN []. Points appearing before the elbow have small distance to their k nearest neighbor and therefore belong to dense cluster interiors. Points that appear after the elbow have substantially larger k distance and are likely to be noise or members of very sparse clusters []. Setting epsilon to the elbow value implements the decision rule: every point whose k distance is smaller than epsilon is considered part of a cluster while points with larger k distance are treated as noise. Points inside clusters tend to have many nearby neighbors so their k distance stays small, and the curve is flat. When the curve reaches the elbow the population of points transitions from cluster interior to boundary or to background. This transition produces the characteristic knee shape that guides epsilon selection. [] Subsequently, the parameter sets that were theoretically derived were validated and fine-tuned empirically using a small representative sample dataset. This process was carried out to ensure the robustness of the application in real environment.

In this project, voxelized point clouds are defined on a discrete integer grid. Distances between voxel centers are computed with the Euclidean norm

$$d = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{1}$$

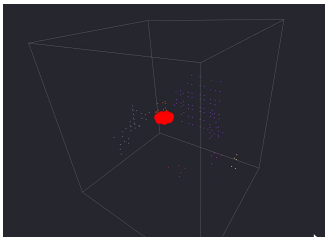
where each delta is an integer difference between voxel coordinates. Because the coordinate differences are integers the set of possible distance values is discrete and limited. When the k distance values for all points are sorted, many points frequently share identical or nearly identical distance values. This results in extended horizontal segments in the sorted curve. Each flat segment corresponds to a bin of identical distance values produced by the grid geometry. When the next larger discrete distance appears, the curve jumps to the next step. The more regular the grid and the coarser the voxel resolution the stronger the staircase effect. The staircase appearance is not a flaw but an artifact of discretization. It implies that small changes to epsilon within a flat segment will not change cluster assignments. Conversely, choosing epsilon values at jump points will change the number of neighbors for many points at once and may cause abrupt changes in the clustering outcome. In practice it is therefore advisable to choose epsilon near the top of a stable flat segment immediately before a jump, or to use algorithms that estimate the elbow by curvature rather than by manual inspection [].

IV. Experimental Results from Testing with Real Sensor Ring Data Input

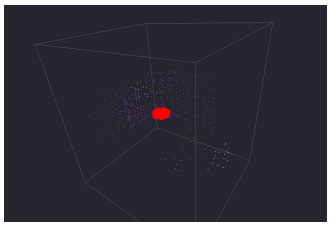
A. Cycle Clock Time

During the testing phase, it became evident that cycle clock time presented a significant constraint, prompting a comprehensive evaluation of both conventional boolean-based voxels and a continuous time-based approach. Theoretically, the boolean method offers enhanced efficiency by reducing complex floating-point arithmetic to low-latency bitwise operations. However, this computational advantage remains to be empirically validated in our specific test environment. It is essential to note that the boolean approach enables more robust movement reconstruction by treating each time frame as a discrete, independent state. In contrast to time-surface methods, which frequently overwrite historical data with the most recent timestamp, this discrete separation method preserves the complete occupancy history. This capability enables significantly more granular and extended retrospective trajectory analysis. Overall, the advantages of the time-based method are more consistent tracking, more adjustability in form of the TIME TOLERANCE parameter and enhanced expandability for possible other sensor rings as there are no time frames but a more fluid memorization of the received sensor data.

After solving the mentioned issues, we were able to obtain results for motion vectors that demonstrated a high level of similarity to the actual movement. However, it should be noted that significant noise interferences occasionally manifested, leading to erratic yet accurate findings (i.e., real direction, but incorrect velocity). Furthermore, DBSCAN occasionally causes spatially distant points to be grouped together when used in environments with a sparse number of voxels. This occurrence can be primarily attributed to the discretization from the voxel-based environment, in combination with the implementation of a density-based approach. Figure 1 clearly shows that the scanner initially detected a pool noodle as a



(a) Case 1: Clear differentiation of objects



(b) Case 2: Merging of Object due to closeness

Fig. 3: Merging phenomena of unrelated objects

separate object. However, when it came close to the wall, the scanner merged both into a single cluster.
Additionally, we observed this issue when the two objects approached each other. As illustrated in Figure X, the present test involved two moving objects observed from a bird's-eye view. The initial grid (1) displays a single individual approaching the sensors at a relatively high velocity, which leads to erratic behavior. As can be seen in grid 2, the movement can be identified with a high degree of accuracy. Grid 3 illustrates the second person's approach toward the sensors. When both subjects are in proximity to the sensors, the detected objects are merged, resulting in an absence or very minimal detected motion. In grid 4, the first person exits the sensor range, and the objects separate once more. The second person's movement in Grid 5 can then be observed with more clarity until he exits the sensor range.
In general, merging reduces our ability to separate objects by distance and therefore to detect different items reliably as well as semantically, hence calculating the proper movement vector. Moreover, initial tests revealed that our sensor ring produces false points, which introduce noise in an already sparse voxel representation. Despite these limitations, we were able to obtain satisfactory results.

B. Impact of Point Density on Voxel-based Occupancy Clustering

XYXY

C. Temporal Object Tracking and Velocity Constraints

Our main program (see: "voxel boolean json" and the other voxel-based-programs) fulfills all of these requirements.
par The initial problem of latency was largely solved. The Raspberry Pi Pico we use to read the data outputs at 8.33 Hz. The main Program need approximately 60 to 70 ms to complete one cycle (which includes data-gathering and all following steps) which was also tested with larger sample sizes from up to 7 sensors.
par A different problem is the fact that there is a highest velocity at which a given object is correctly identified. The problem originates from the way the main program tries to trail the objects. To discretize the problem we simply group the data recieved by the sensors together by time. It seems best to use the time needed for one program cycle for this "clock frequency", that means 0.15 seconds or 6.67 Hz in our program. Fundamentally, there are now two differing possible methods: By checking for overlap between each time-frame we can track each object over time. This is very reliable for low velocities and efficient, because it can be expressed as simple repeated matrix-multiplications. As such it is our main tracking method. As long as the object velocity is smaller than the exposed lenght (meaning the length of the object perpendicular to the detecting sensor) of the object divided by the clock frequency the program will detect overlapping voxels, but above this velocity the object will likely have already moved out of the way. The Program will then regard the object as two seperate objects, with a velocity of zero (see example image below).

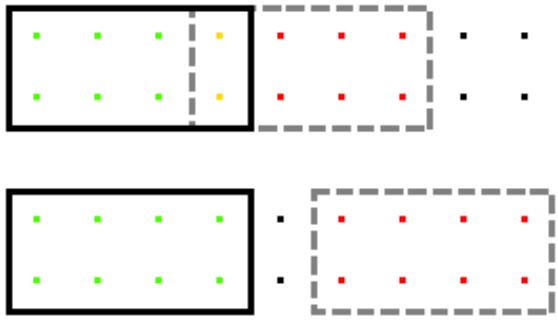


Fig. 4: Overlapping Technique of Tracking Objects

- Grey boxes: position of the object at t=0
- Black boxes: position of the object at t=1
- Red dots: active voxels at t=0
- Green dots: active voxels at t=1
- Yellow dots: voxels which are active at t=0 and t=1
- Black dots: inactive voxels

The upper example shows a case with a velocity lower than the maximum detecable, the lower with a velocity higher.
To complement this, one could also search in proximity of the detected object, but this is disproportionately inefficient as now the program would have to actually compute the distances. One way we could solve these performance disadvantages is by using a method called morphological dilation. Instead of comparing the distance between points, one objects volume is virtually expanded in size by the radius we would otherwise check the distance for. In the next step, we simply have to once again search for an overlap. This way is not only more efficient, but also far more consistent for different object sizes. This way however does not enable to detect velocities larger than the searched radius divided by cycle time, but cooperates nicely with our previously described main tracking method. A different approach could be to limit the searched volume, for example by using previously determined movement-vectors to restrict the direction. The problem arising from this is the fact that one would need a first movement-direction to begin

with, meaning it would have to have already been observed and then accelerated, or have just entered the workspace with an unkown velocity, but likely with a direction into the workspace. This last case (fast object entering the workspace) is arguabely the most dangerous, but also the most difficult to detect and track.

Another problem stems from the inherent uncertainty of the voxel-solution. The further away an object is from the sensor, the more voxels are availabe to represent each data-input. This creates two critical regions: At a certain distance gaps begin to form between the voxels, creating unclear geometries. This can be fixed by increasing the amount of activated voxels with increasing distance to the sensor. The other critical region is the region directly in front of the sensor. Objects in close proximity can never be fully represented in the same voxel size and will activate most voxels directly in front of the sensor which results in erratic movement detection. Two possible solutions for this problem are to either increase the voxel-density in the proximity around the sensors (for example by using Adaptive Mesh Refinement) or by using data from a different sensor whose view of the critial region is unobstructed, both of which exceed the scope of this project.

V. Limitations

Fundamentally there are two forms of limitations for our final program: Theoretical limitations which stem from the way our approach is designed and real-life limitations which mainly come from uncertainties of the used sensors.

The system's detection capabilities are fundamentally constrained by the target's effective profile, which consists of distance, relative velocity, and material surface properties (ex.g. signal absorption in translucent materials or scattering in highly reflective ones) []. Empirical findings revealed a steep decline in detection efficacy. Low-profile geometries (e.g., 2cm diameter rods) were detectable only in immediate proximity, whereas targets exceeding a 5cm characteristic diameter (e.g., human limbs) maintained consistent detectability at ranges surpassing 100cm. Furthermore, dynamic targets demonstrated a significantly higher probability of detection compared to static counterparts. This enhancement is attributed to increased spatial sampling, as moving objects traverse multiple sensor fields of view (FOV), thereby accumulating more data points. However, deriving a precise velocity-based threshold remains challenging due to the complex correlation between varying spatial precision and signal quality across different distances.

VI. Conclusion

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] Q. Hallez and E. Bouzbib, "Human-Cobot collaboration's impact on success, time completion, errors, workload, gestures and acceptability during an assembly task", *arXiv preprint arXiv:2405.17910*, 2024. [Online]. Available: https://arxiv.org/abs/2405.17910

[5] 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

[6] 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

[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.