Stochastic 2D LiDAR-Camera Sensor Fusion for Real-Time Object Mapping and Tracking

by

Thijs Domburg

Student Number:	5153433
Project Duration:	January, 2024 - November, 2024
Faculty:	Mechanical Engineering (ME)
Department:	Cognitive Robotics (CoR)
Supervisors:	T. Dalhuisen (Dalco Robotics)
	Prof. dr. ir. M. Wisse (TU Delft)
Graduation Committee:	Prof. dr. ir. M. Wisse (TU Delft, Chair)
	T. Dalhuisen (Dalco Robotics)
	Dr. M. Kok (TU Delft)
	Dr. G. Chen (TU Delft)

TU Delft

DALCO ROBOTICS

Stochastic 2D LiDAR-Camera Sensor Fusion for Real-Time Object Mapping and Tracking

Thijs Domburg (5153433)
Delft University of Technology - Cognitive Robotics

Dalco Robotics Supervisor: Tom Dalhuisen
TU Delft Supervisor: Martijn Wisse

October 23, 2024

***Abstract*—Spatial object detection and environmental understanding are fundamental aspects of autonomous driving in mobile robots. In this work, a stochastic approach to 2D LiDAR and stereo camera sensor fusion for object detection is presented. By tracking LiDAR clusters 360° around the robot and continuously updating knowledge of their corresponding object types using Bayesian inference when they are within the camera's view, uncertainties in the detection algorithm are addressed. Additionally, localisation and sensor uncertainties are accounted for by using Gaussian distributions to model the locations of entities. By estimating the state of these entities, dynamic objects can be tracked throughout the environment, while remembering their type from when they were last visible in the camera's view. The results demonstrate correct linking of camera detections with LiDAR clusters, with an average positional inaccuracy of objects of 0.12m in simulation, and 0.25m across different experiments on the robot, as well as the ability to track humans even when they move out of the camera's field of view with an average tracking error of 0.33m in simulation.**

I. Introduction

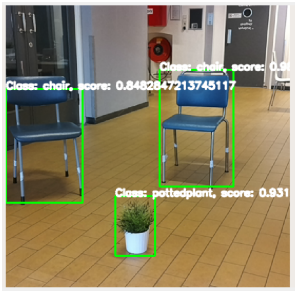
Accurate spatial object detection and environment understanding are fundamental for autonomous driving in mobile robots. For a robot to navigate and interact safely and effectively in dynamic environments, it must accurately perceive and model its surroundings. This involves not only detecting objects, but also classifying their types and tracking their movements over time.

The *Rober* robot [1] is such a mobile robot, designed to transport food in restaurants, while navigating dynamic spaces. It is equipped with a 2D LiDAR at the bottom, and a stereo camera at the front, as shown in Figure 2.

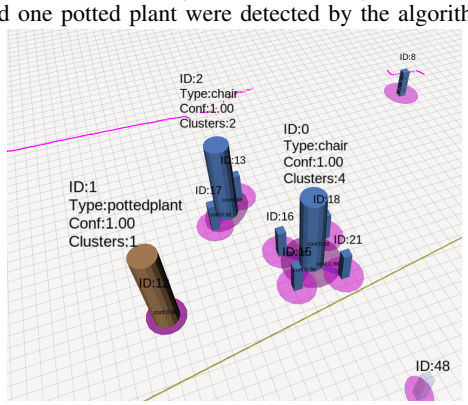
The LiDAR sensor provides high-resolution spatial data for detecting objects and obstacles but lacks information on their identity or type. To complement the spatial awareness provided by LiDAR, the stereo camera captures visual data, allowing object classification based on appearance. However, these sensors are not 100% accurate due to noise and inaccuracies, which should be taken into account. Additionally, the robot's self-localisation is not always precise.

One important aspect of objects in the environment of a robot is persistence: confidence in an object's presence should increase as more evidence accumulates over time. Additionally, once detected, the system should maintain awareness of their presence, even if they temporarily move out of sight. [2]

This paper presents a ROS2-based system that integrates the strengths of LiDAR and camera technologies to perform object detection, classification, and tracking for autonomous mobile robots. The system is designed to address real-world uncertainties, including sensor noise and localisation



(a) YOLO detections of objects in the testing environment. Two chairs and one potted plant were detected by the algorithm.



(b) Mapped objects by the proposed system. Small cuboids are clusters in the LiDAR data, while the tall cylinders are objects detected by the camera. The LiDAR clusters are correctly linked to the detected objects. The purple circles represent the uncertainty in the location of the entities.

Fig. 1: Modeled environment at the bottom, and the image from the camera with the object detections at the top.

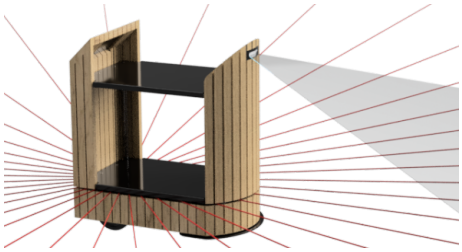


Fig. 2: The robot used in the experiments. It has a 2D LiDAR at the bottom, and a stereo camera at the front. [1]

inaccuracies. In this system, LiDAR data is clustered and tracked over time, with clusters being associated with objects detected by the YOLO algorithm [3] on the camera. Through Bayesian inference, the system continuously updates these associations, enabling persistent object tracking even when objects move outside the camera's field of view. This is particularly valuable for tracking moving objects, such as humans, and ensures continuous object awareness. By combining data from both sensors, the system constructs a comprehensive environmental model that integrates spatial layout with object semantics. Figure 1 provides a visualisation of the system's output, showing correctly linked objects and clusters.

In section II the related work is discussed, followed by an overview of the method in section III. Key aspects of the model are further explained in the sections IV, V and section VI. The performance is tested in section VII, and the results are discussed and concluded in sections VIII and IX respectively.

II. Related Work

Modelling an environment has always been a key aspect of autonomous robotics. Most robots still rely on geometric maps of static environments for navigation and localisation. However, dynamic or changing environments pose challenges to these approaches. In recent years, significant progress has been made toward giving robots a deeper understanding of their environment.

Object-level Mapping with Camera: While numerous methods exist for recreating environments by modeling the mesh of objects ([4], [5], [6]), more computationally efficient approaches focus on tracking the position and orientation of objects, rather than their full shape. Slam++ [7] achieves this by detecting and modeling prescanned objects, avoiding the need to compute object meshes for each instance. Simpler methods rely on 2D camera images to detect objects, as seen in the work of Dong et al. [2], who used YOLO to detect objects and employed Bayesian inference to increase confidence in their presence as they appeared in multiple frames. This approach also retained memory of objects even when they became occluded or moved out of view. However, dynamic objects were treated as outliers in their work. CubeSLAM [8] also uses YOLO and accounts for dynamic objects by assuming constant motion, while DynaSLAM II [9] and VDO-SLAM [10] use custom detection algorithms to track objects by comparing visual features across images using optical flow, although they achieve limited tracking accuracy due to the inaccuracy of camera depth estimation.

Object-level Mapping with 2D LiDAR: Although many methods exist for object detection using 3D LiDAR, research on object detection with 2D LiDAR is less common, as planar data alone is insufficient to determine object types. Kwon et al. [11] developed a stochastic map building method in 1999 that could handle quasi-static environments. By clustering 2D LiDAR points and parameterising them stochastically, they were able to model the environment and update it by comparing new data with the existing model. Similarly, [12] performed circle fitting on clusters to identify objects in agricultural environments and updated their positions over time. However, both methods are limited by their inability to handle dynamic objects. [13] also attempts to find circular poles in vineyards but also accounts for dynamic objects, using an analytical approach instead of Euclidean clustering. This analytical approach is also described in [14].

In the context of autonomous vehicles on the road, the approach in [15] addresses dynamic objects by detecting vehicles, fitting L shapes (from the two visible sides of a vehicle) on clustered data, and associating detected objects using a global nearest neighbor algorithm. However, this system is restricted to tracking vehicle shapes. The work in [16] can handle other shapes on the road by modeling objects with a set of sample points along their boundaries and updating them with raw LiDAR data. This approach is computationally more intensive, and the data association is more challenging.

Object-level LiDAR Camera Fusion: Numerous systems perform 3D LiDAR and camera bounding box fusion. Some methods detect objects using the camera image, after which the 3D position of the object is estimated using the LiDAR [17]. Approaches discussed in [18] and [19] enhance the LiDAR pointcloud with camera information to perform detection using the combined 3D pointcloud. Additionally, deep learning networks are used to fuse LiDAR clouds with images to compute 3D object positions [20], [21].

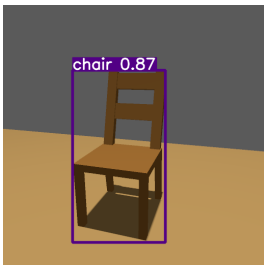
Research on 2D LiDAR-camera fusion is relatively limited. Mulyanto et. al [22], and more recently Hwang et. al [23], presented a system that fused camera and 2D LiDAR measurements to improve distance measurements to objects in each frame.

Most methods for LiDAR-camera fusion process the data on a per-frame basis. Although these methods can achieve high accuracy for individual frames, they do not account for time when fusing LiDAR data with camera images. Incorporating data from multiple timestamps can improve the robustness of sensor fusion, making it less susceptible

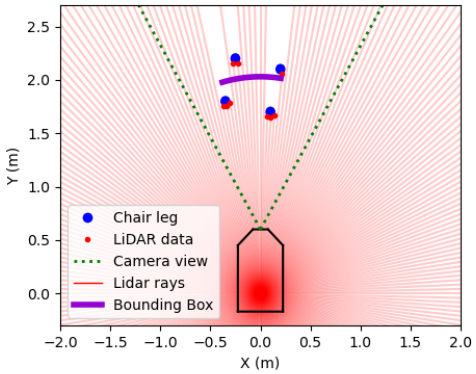
to noise and sensor inaccuracies.

III. System Overview

The *Rober* robot, used in the experiments (see Figure 2), is equipped with a 2D RPLIDAR S2 360° laser scanner at the bottom and an OAK-D Pro FF stereo camera at the front. With YOLOv4 [3] running on the camera, objects are detected with bounding boxes and depth estimations, derived by analysing the stereo point cloud from the camera. This process is further explained in appendix B. Additionally, the LiDAR detects parts of objects close to the floor, such as chair- or human legs. The visibility of the robot's sensors are visualised in Figure 3, where the YOLO detection of the chair is displayed in the horizontal plane, alongside the LiDAR measurements of its legs in Figure 3b.



(a) YOLO detection of a chair in front of the robot.



(b) Visibility of the robot from above. The LiDAR (red) detects the 4 legs of a chair (blue) as separate clusters, while the camera detects the chair on the image with a bounding box, mapped to a horizontal plane with a purple line. The green dotted lines represent the camera's field of view.

Fig. 3: Sensor measurements from the robot. (a) shows YOLO's object detection on the camera image, and (b) displays both the projected bounding box and LiDAR measurements on the horizontal plane.

The LiDAR data is first clustered using the *EuclideanClusterExtraction* algorithm from the Point Cloud Library (PCL) [24], which implements a Kd-tree structure for finding the nearest neighbors of LiDAR points, inspired by [25]. Clustering the pointcloud greatly reduces the computational load of the system, while allowing it to process all detected entities. The goal is to connect the LiDAR clusters to their corresponding objects detected by the camera, to increase the positional accuracy of the objects, and to enable object tracking outside camera's field of view.

The resulting environment model consists of two types of entities: clusters and objects.

Clusters represent entities detected by the 2D lidar. Each cluster has a global position on the map, which is updated whenever the cluster is visible in the LiDAR data. Newly detected clusters are compared with existing ones in the environment model to find matches, allowing the model to be updated accordingly. Moving clusters are tracked over time by updating their positions using Bayesian inference.

Objects are labeled entities composed of one or multiple clusters. For example, the four legs of a chair are visible in LiDAR data as separate clusters, though they belong to the same object, the chair. Confidence in the association between clusters and objects grows over time as more evidence from the camera becomes available. Object labels are updated when they are visible to the camera, while object positions are also updated based on LiDAR clusters outside the camera's view.

The system operates at 10 Hz and has two main steps. The first step involves receiving and preparing new measurements from the LiDAR and camera. Positional uncertainties in these measurements are determined based on sensor inaccuracies and robot localisation uncertainties, as explained in section IV.

The second step involves associating these new measurements with existing data when applicable, and updating the environment. This is discussed in section V.

IV. Measurement Uncertainty

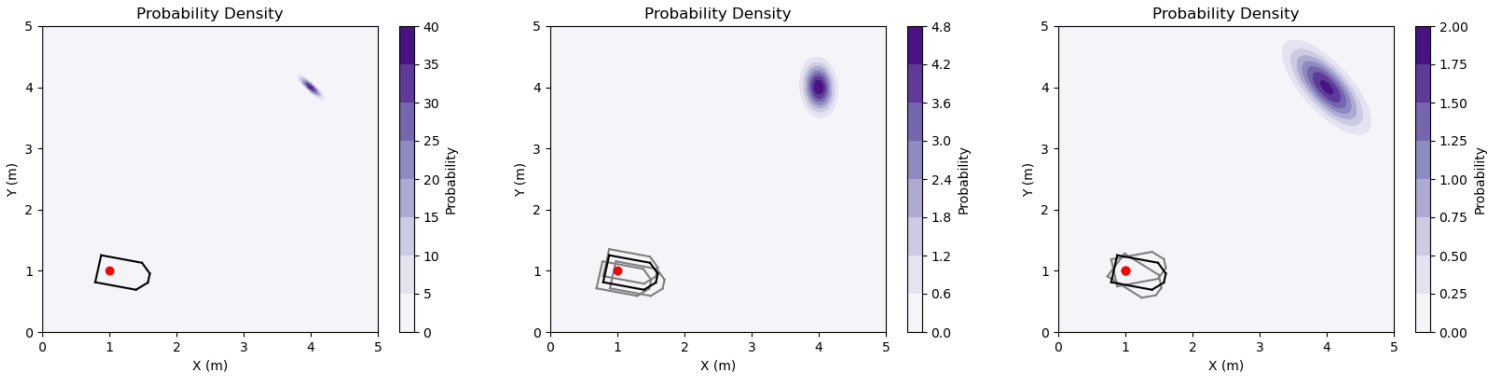
The first step in the process is to gather new data. As mentioned earlier, the system uses both a LiDAR and a stereo camera, which deliver their data independently.

A. LiDAR data

After the LiDAR data is clustered, each cluster has a position in the world frame. However, there are uncertainties regarding the exact position. The positional uncertainties of the clusters are due to both sensor noise and localisation inaccuracy. The measured position M_C of a cluster can be modeled as a function of the cluster's true position P_C, with added noise from the sensor and localisation, expressed as:

$$M_C = P_C + \text{Noise}_{S_{\text{LiDAR}}} + \text{Noise}_L, \tag{1}$$

where both noise terms are assumed to follow a zero-mean Gaussian distribution with covariances $\Sigma_{S_{\text{LiDAR}}}$ and Σ_L



(a) Probability distribution of a LiDAR cluster in map frame with sensor uncertainty.

(b) Probability distribution of a LiDAR cluster in map frame with sensor uncertainty and positional localisation uncertainty.

(c) Probability distribution of a LiDAR cluster in map frame with sensor uncertainty, positional localisation uncertainty and rotational localisation uncertainty.

Fig. 4: Probability distribution of a cluster in map frame with the uncertainties added step by step. The uncertainties implemented are the sensor uncertainty (a), positional localisation uncertainty (b) and rotational localisation uncertainty (c). The LiDAR is visualized in red.

respectively.

The **LiDAR sensor uncertainty** can be decomposed into angular and distance uncertainties, which are first defined in polar coordinates and then transformed to Cartesian coordinates using the Jacobian matrix:

$$\Sigma^{\text{cart}}_{S_{\text{LiDAR}}} = J \Sigma^{\text{polar}}_{S_{\text{LiDAR}}} J^T \tag{2}$$

With

$$J = \begin{bmatrix} \cos(\theta) & -r\sin(\theta) \\ \sin(\theta) & r\cos(\theta) \end{bmatrix} \tag{3}$$

$$\Sigma^{\text{polar}}_{S_{\text{LiDAR}}} = \begin{bmatrix} \sigma^2_{r_{\text{LiDAR}}} & 0 \\ 0 & \sigma^2_{\theta_{\text{LiDAR}}} \end{bmatrix} \tag{4}$$

where θ and r are the angle and distance of the LiDAR cluster relative to the sensor, and $\sigma^2_{r_{\text{LiDAR}}}$ and $\sigma^2_{\theta_{\text{LiDAR}}}$ are distance and angular covariances of the LiDAR, pre-set by the user or manufacturer. In the tests, the angular standard deviation $\sigma_{\theta_{\text{LiDAR}}}$ is set at 0.05, which is a combination of the angular resolution of the LiDAR and additional uncertainty in the LiDAR's mounting on the robot. Due to divergence of the light beam of the LiDAR, the uncertainty in the distance of the measurement also depends on the distance itself. In the tests, the distance standard deviation $\sigma_{r_{\text{LiDAR}}}$ is set at $0.05 + 0.01d$, with d being the distance of the measurement to the sensor. Similar to the angular measurement, this is a combined uncertainty of the value specified by the manufacturer (0.03m [26]) and uncertainty in the exact position of the LiDAR on the robot. In Figure 4a, this sensor uncertainty is visualised.

The **localisation uncertainty** consists of both positional and rotational components. These are calculated by the AMCL (Adaptive Monte Carlo Localisation) algorithm of nav2 [27], which is used for localisation of the robot. The positional x- and y-axis uncertainties are represented by the following components of the localisation covariance matrix:

$$\Sigma_{L_{\text{pos}}} = \begin{bmatrix} \sigma^2_{xx_{\text{AMCL}}} & \sigma^2_{xy_{\text{AMCL}}} \\ \sigma^2_{yx_{\text{AMCL}}} & \sigma^2_{yy_{\text{AMCL}}} \end{bmatrix} \tag{5}$$

The addition of this uncertainty is visualised in Figure 4b. The rotational uncertainty is computed similarly to the LiDAR sensor uncertainty but involves only the rotational component coming from the localisation algorithm:

$$\Sigma^{\text{cart}}_{L_{\text{rot}}} = J \Sigma^{\text{polar}}_{L_{\text{rot}}} J^T \tag{6}$$

With J equal to Equation 3 and the uncertainty matrix given by:

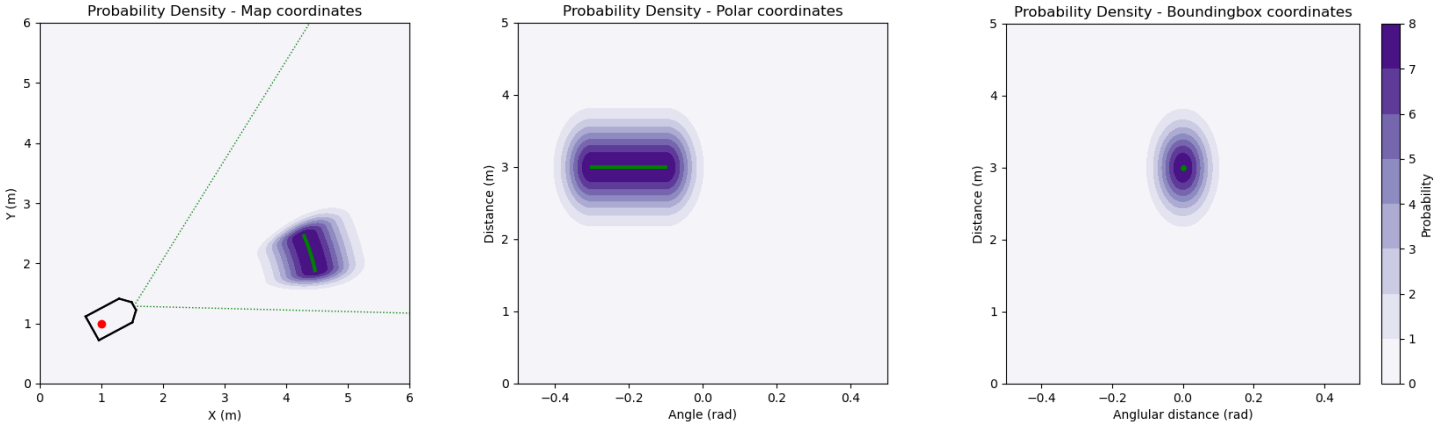
$$\Sigma^{\text{polar}}_{L_{\text{rot}}} = \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2_{\theta_{\text{AMCL}}} \end{bmatrix} \tag{7}$$

The total localisation uncertainty is the sum of both positional and rotational uncertainties:

$$\Sigma_L = \Sigma_{L_{\text{pos}}} + \Sigma^{\text{cart}}_{L_{\text{rot}}} \tag{8}$$

Finally, the total uncertainty of the cluster's location is a combination of the sensor and localisation uncertainties, resulting in a Cartesian normal distribution with the mean at the measurement position M_C and a combination of the sensor and localisation noise: $\mathcal{N}^{\text{cart}}(M^{\text{cart}}_C, \Sigma^{\text{cart}}_{S_{\text{LiDAR}}} + \Sigma^{\text{cart}}_L)$.

This normal distribution consisting of all the uncertainties is visualised in Figure 4c.



(a) Probability distribution of an object detected with a bounding box in map frame. The camera field of view is visualised with green dotted lines, while the bounding box is visualised with the solid line in the middle of the distribution.

(b) Probability distribution of an object detected with a bounding box in polar coordinates, with the camera being the center of the coordinate system. The bounding box, which is a straight line in polar coordinates, is visualised in green.

(c) Probability distribution of an object detected with a bounding box in bounding box frame. The x-axis represents the angular distance to the closest part of the bounding box.

Fig. 5: Probability distribution of bounding box detection in (a) map frame, (b) polar camera frame, and (c) bounding box frame. The bounding box frame is similar to the polar coordinate system. However, the angular component represents the angular distance to the bounding box.

B. Stereo camera data

Similar to the cluster data from the LiDAR, each detection from the camera has its global position based on the depth estimate of the stereo camera. This position is calculated by analysing the density of the part of the stereo pointcloud that falls within the bounding box, as further explained in Appendix B. The measured position M_D of a detection is related to the true position of the object P_O by incorporating the sensor and the localisation noise, which can be expressed as:

$$M_D = P_O + \text{Noise}_{S_{\text{Camera}}} + \text{Noise}_L, \tag{9}$$

Unlike LiDAR, where entities are represented by specific x and y coordinates, camera detections are represented by bounding boxes with a certain width. Therefore, instead of modeling this as a Gaussian distribution in Cartesian coordinates, the distribution must be adjusted so that all coordinates within the bounding box have high probability density, with the probability rapidly decaying outside the box. This visualised in Figure 5a.

To achieve this, the the coordinates are transformed to polar coordinates, with the camera position as the origin, using the following equations:

$$r = \sqrt{x^2 + y^2} \tag{10}$$

$$\theta = \arctan(\frac{y}{x}) \tag{11}$$

where x and y are the Cartesian coordinates relative to the camera. The probability distribution in polar coordinates is visualised in Figure 5b.

Rather than directly using the result of this transformation, the polar coordinates are converted into a bounding-box coordinate system. This coordinate system is unique for all detections. In this system, the angular component represents the angular distance to the bounding box. Any coordinate that falls within the bounding box in polar coordinates has a distance of zero, while points outside the bounding box have a value equal to the distance to the nearest edge of the box. This is visualised in Figure 5c and done using the following equations:

$$\begin{cases} \theta^{\text{polar}} > \theta^{\text{polar}}_{\text{bbox\_max}}, & \theta^{\text{bbox}} = \theta^{\text{polar}} - \theta^{\text{polar}}_{\text{bbox\_max}} \\ \theta^{\text{polar}} < \theta^{\text{polar}}_{\text{bbox\_min}}, & \theta^{\text{bbox}} = \theta^{\text{polar}} - \theta^{\text{polar}}_{\text{bbox\_min}} \\ \text{else}, & \theta^{\text{bbox}} = 0.0 \end{cases} \tag{12}$$

where $\theta_{\text{bbox\_max}}$ and $\theta_{\text{bbox\_min}}$ are the angular coordinates of the edges of the bounding box.

In this bounding-box coordinate system, the **camera sensor uncertainties** in both the angular and distance directions can be defined as a Gaussian distribution with covariance:

$$\Sigma^{\text{bbox}}_{S_{\text{camera}}} = \begin{bmatrix} \sigma^2_{r_{\text{camera}}} & 0 \\ 0 & \sigma^2_{\theta_{\text{camera}}} \end{bmatrix} \tag{13}$$

In the tests, the angular standard deviation $\sigma_{\theta_{\text{camera}}}$ is set at 0.05. Given the limited accuracy of depth estimation using stereo cameras, the distance standard deviation $\sigma_{r_{\text{camera}}}$ is set at $0.25 + 0.15d$, where d is the distance of the detection to the sensor. This setup is similar to the uncertainty model for the LiDAR sensor. The base uncertainty of 0.25m accounts

for the fact that objects have a size, and the centroid of the visible part of the object may not align with the actual centroid. For example, a large portion of a chair visible to the camera may be its backrest, which is offset from the object's centroid. Using this base uncertainty helps ensure that elements like chair legs, which are not at the object's centroid, are correctly associated with the object. The additional uncertainty of 0.15m for each meter further from the camera, is due to the rapid decrease in depth point cloud accuracy. Experimental comparisons between the true distance and the camera-detected distance revealed an approximate increase in distance error of 15 cm per meter.

Adding the **localisation uncertainty** follows a procedure similar to that used for the new LiDAR clusters. However, since the detection is in the bounding box coordinate system, the localisation uncertainties must also be transformed. The total localisation uncertainty becomes:

$$\Sigma_L^{\text{bbox}} = \Sigma_{L_{\text{pos}}}^{\text{bbox}} + \Sigma_{L_{\text{rot}}}^{\text{bbox}} \tag{14}$$

where $\Sigma_{L_{\text{pos}}}^{\text{bbox}}$ and $\Sigma_{L_{\text{rot}}}^{\text{bbox}}$ represent the positional and rotational uncertainty of the localisation algorithm in bounding-box coordinates. Since the bounding-box coordinate system is derived from the polar coordinate system, these uncertainties remain the same in polar coordinates, simplifying the calculations. The positional uncertainty of the localisation algorithm is initially provided in Cartesian coordinates and can be transformed using the following equation:

$$\Sigma_{L_{\text{pos}}}^{\text{bbox}} = \Sigma_{L_{\text{pos}}}^{\text{polar}} = J\Sigma_{L_{\text{pos}}}^{\text{cart}} J^T \tag{15}$$

Where

$$J = \begin{bmatrix} \frac{x}{r} & \frac{y}{r} \\ -\frac{y}{r^2} & \frac{x}{r^2} \end{bmatrix} \tag{16}$$

$$\Sigma_{L_{\text{pos}}}^{\text{cart}} = \begin{bmatrix} \sigma^2_{xx_{\text{AMCL}}} & \sigma^2_{xy_{\text{AMCL}}} \\ \sigma^2_{yx_{\text{AMCL}}} & \sigma^2_{yy_{\text{AMCL}}} \end{bmatrix} \tag{17}$$

where x and y are the position of the detection in cartesian coordinates, and r and θ in polar coordinates. The rotational uncertainty can be simply given by:

$$\Sigma_{L_{\text{rot}}}^{\text{bbox}} = \Sigma_{L_{\text{rot}}}^{\text{polar}} = \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2_{\theta_{\text{AMCL}}} \end{bmatrix} \tag{18}$$

This results in the final positional uncertainty, represented by a normal distribution: $\mathcal{N}^{\text{bbox}}(M_D^{\text{bbox}}, \Sigma_{S_{\text{camera}}}^{\text{bbox}} + \Sigma_L^{\text{bbox}})$.

In addition to positional uncertainty, each detection is associated with an object type and a corresponding confidence score.

V. Update Environment Model

As explained in section III, the second step in the process involves associating new measurements with existing entities in the environment model, and updating the model. There are three main aspects that need to be updated in the environment model:

1) **Clusters:** As cluster positions are derived from measurements with uncertainty, the clusters in the environment also carry positional uncertainty. After associating new measurements with existing clusters, the cluster's position and its uncertainty should be updated. Clusters are tracked in the environment, with their positional uncertainty and existence probability updated using bayesian inference based on new measurements.
2) **Objects:** Objects are updated with new bounding-box detections from the camera. After associating the bounding boxes with existing objects, the object's type probabilities are updated using Bayesian inference. Similarly to clusters, their existence probability is updated based on new detections.
3) **Object-cluster connections:** Each cluster has a set of potential parent objects with associated probabilities. These probabilities are updated using the information of the new detections. The likelihood that a cluster belongs to an object is computed based on their relative position to the detection.

A. Update clusters

Before updating the clusters, new measurements must be associated with existing data. To match new measurements with existing clusters, a nearest neighbor association technique using the Mahalanobis distance is employed. [28]

The Mahalanobis distance measures the distance between two points in a multivariate space, considering the covariance of the distribution rather than relying on the simple Euclidean distance. The Mahalanobis distance between a point and a distribution is given by [29]:

$$D_m(\mathcal{N}(\vec{\mu}, \Sigma), \vec{p}) = \sqrt{(\vec{p} - \vec{\mu})^T (\Sigma)^{-1} (\vec{p} - \vec{\mu})} \tag{19}$$

Since both the new measurement and the existing cluster have covariance matrices, the Mahalanobis distance between two Gaussian distributions can be expanded to account for this. This equation is also used in track-to-track fusion for multiple sensors and is given by [30]:

$$D_m(\mathcal{N}(\vec{\mu}_1, \Sigma_1), \mathcal{N}(\vec{\mu}_2, \Sigma_2)) = \sqrt{(\vec{\mu}_1 - \vec{\mu}_2)^T (\Sigma_1 + \Sigma_2)^{-1} (\vec{\mu}_1 - \vec{\mu}_2)} \tag{20}$$

To compute the likelihood that measurement M_i is associated with an existing cluster C_j, the Mahalanobis distance can be used in the following probability function:

$$P(M_i \to C_j) = \exp(-\frac{D_m^2}{2}) \tag{21}$$

For each measured cluster, the likelihood that it corresponds to each existing cluster is calculated, forming

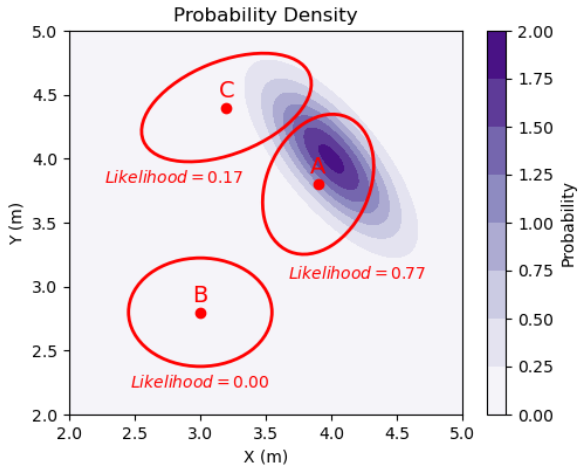


Fig. 6: Likelihoods of hypotheses that existing clusters A, B and C are associated to a single measurement. The clusters are visualised with their covariance in red. while the distribution of the new measurement is visualised in purple.

hypotheses. Examples of these hypotheses are visualised in Figure 6, where a single measurement is compared to multiple existing clusters in proximity. As some existing clusters will not be visible in the current LiDAR scan, and new clusters will be discovered, a likelihood threshold is used to avoid mismatches between these, as also mentioned in [30]. Hypotheses are stored if their likelihood exceeds this threshold. Once all new clusters are evaluated, the list of hypotheses is sorted by likelihood, and potential matches are processed starting with the highest likelihood.

Starting with the hypothesis with the highest likelihood, the cluster is updated with the measurement data, and both the new cluster and the existing cluster are marked as "used" to prevent further associations. Subsequent hypotheses involving already matched clusters are ignored, and the next hypothesis is evaluated. This continues until all matches are processed and unmatched measurements are treated as newly detected clusters. This process is given in Algorithm 1.

For each successful association, the existing cluster with positional probability $\mathcal{N}(\vec{\mu}_{\text{prior}}, \Sigma_{\text{prior}})$ is updated with the measurement with positional probability $\mathcal{N}(\vec{\mu}_{\text{meas}}, \Sigma_{\text{meas}})$. The parameters of the resulting posterior Gaussian distribution $\mathcal{N}(\vec{\mu}_{\text{posterior}}, \Sigma_{\text{posterior}})$ are computed with the following equations, which are the information form of the the update step of a Kalman filter [31].

$$\Sigma_{\text{posterior}} = (\Sigma_{\text{prior}}^{-1} + \Sigma_{\text{meas}}^{-1})^{-1} \tag{22}$$

$$\vec{\mu}_{\text{posterior}} = \Sigma_{\text{posterior}}(\Sigma_{\text{prior}}^{-1}\vec{\mu}_{\text{prior}} + \Sigma_{\text{meas}}^{-1}\vec{\mu}_{\text{meas}}) \tag{23}$$

This update is visualised in Figure 7, where the old distribution in light red is updated with the measurement, resulting in the new distribution in darker red.

Algorithm 1 Process association hypotheses

define $measurements$ = new cluster measurements
define $clusters$ = existing clusters
define $association\_hypotheses$ = association hypotheses

procedure PROCESS HYPOTHESES($association\_hypotheses$, $measurements$, $clusters$)
  for $hypothesis$ **in** $association\_hypotheses$ **do**
    if $hypothesis.cluster.updated$ **then**
      continue
    end if
    if $hypothesis.measurement.used$ **then**
      continue
    end if
    Update $cluster$ with $measurement$
    $hypothesis.measurement.used \leftarrow True$
    $hypothesis.cluster.updated \leftarrow True$
  end for
end procedure

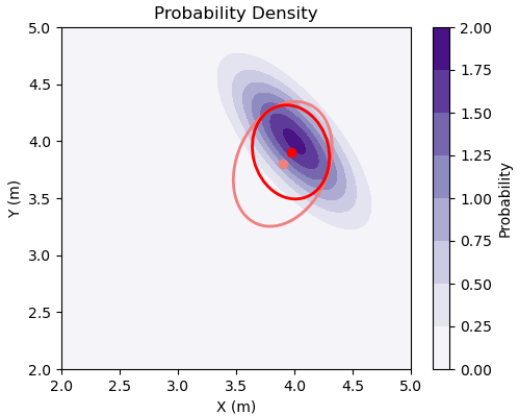


Fig. 7: Updating the positional probability of a cluster with a new measurement. The old distribution is visualised in light red, and the new distribution in dark red. The position distribution of the measurement is shown in purple.

As a cluster is updated, confidence in its existence should increase. Bayesian updates are applied, where the probability that a hypothesis H (that a cluster exists) is true, given evidence E, should increase when an object is just seen, and decrease when no information is given. More about these update rules and the values used in specific situations is given in Appendix C.

At each time step, new positions of the clusters are predicted based on their last known velocity. Additional process noise is incorporated to account for the uncertainty in this prediction. For static clusters, this noise is modeled as a zero-mean Gaussian distribution with covariances of 0.02 in the x- and y-directions. For dynamic clusters, additional uncertainty is included in the direction of their velocity, with a velocity factor of 0.2 for the longitudinal direction of the velocity and 0.05 for the lateral direction.

This prediction allows the system to continue tracking clusters even when they are temporarily occluded.

B. Update objects

For object detections, a slightly different approach is required. As explained earlier, the positional uncertainty of the camera detections is modelled as a Gaussian distribution in the bounding-box coordinate system. To associate new detections with existing objects, the positional uncertainty of the existing objects must also be transformed into the bounding box coordinate space. Once this transformation is complete, the Mahalanobis distance between the two Gaussian distributions can be calculated using Equation 20. Examples of this positional likelihood are visualised in Figure 8.

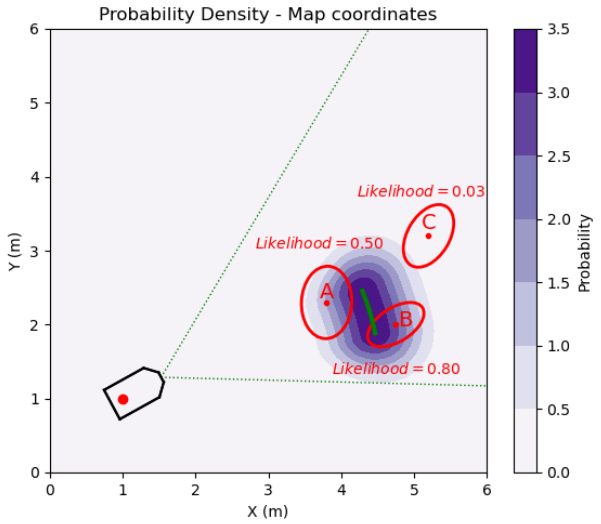


Fig. 8: Positional likelihoods of hypotheses that objects A, B and C are associated to the detection. The existing entities are visualised with their covariances in red, while the distribution of the new detection is visualised in purple. The bounding box is visualised by a green line.

In addition to the distance, the object type is considered in the association process. If a newly detected chair appears in the same location as a previously detected table, it is less likely to be the same object compared to a situation where a chair had been detected there earlier. This type similarity is achieved by multiplying the confidence of the detection with the confidence of the object being the same type as the detection. For example, if a chair is detected with a probability of 0.9 and compared to an object with a probability of 0.7 of being a chair and 0.3 of being a table, the resulting type similarity probability is $0.9 \cdot 0.7 = 0.63$. If the detection were a table with a confidence of 0.9, the similarity probability would be $0.9 \cdot 0.3 = 0.27$, indicating a lower likelihood that it is the same object. However, some objects are more similar than others, making it easier to confuse them. Further discussion about this is provided in section VIII.

The likelihood that a new detection D_i corresponds to existing object O_j can be computed using a combination of the positional and type likelihoods:

$$P(D_i \to O_j) = \exp(-\frac{D_m^2}{2}) \cdot (P(O_j^{\text{type}} = D_i^{\text{type}}) P(D_i^{\text{type}})) \quad (24)$$

Using both the positional distribution probability and the type similarity probability, the same process used for cluster association is applied to associate new detections with existing objects.

For each successful association between an existing object and a YOLO detection, the object's stored information is updated. Each object in the environment model has a set of possible types, each with an associated probability, normalized so that the total probability sums to 1. Initially, the object is assigned the type "unknown" with a probability of 1.0. When new data (a YOLO detection) is collected, the detected type is either updated if it already exists in the object's type list, or added as a new type if it does not exist, with an initial probability of 0.1. The updated probability for each type is calculated using Bayesian inference, and then all probabilities are normalized again to ensure they sum to 1. This process is detailed in Algorithm 2.

Algorithm 2 Update object type

define *detection_type* = type of YOLO detection
define *confidence* = confidence of YOLO detection
define *class_hypotheses* = list of classes with probabilities

unknown_hypothesis.type ← −1
unknown_hypothesis.probability ← 1.0
Add *unknown_hypothesis* to *class_hypotheses*

procedure UPDATE TYPE PROBABILITIES(*detection_type*, *confidence*)
  for *hypothesis* **in** *class_hypotheses* **do**
    if *hypothesis.type* = *detection_type* **then**
      Update *hypothesis.probability* with *confidence*
      Normalise probabilities
      return
    end if
  end for
  new_hypothesis.type ← *detection_type*
  new_hypothesis.probability ← 0.1
  Update *hypothesis.probability* with *confidence*
  Add *new_hypothesis* to *class_hypotheses*
  Normalise probabilities
  return
end procedure

Since the LiDAR provides more accurate positional data, the positional uncertainty of objects depends on their associated clusters, if any exist. The *xy*-spread and average uncertainty of the child clusters are used as the object's positional uncertainty. Similar to the existence probability of clusters, Bayesian updates are applied to update the existence probability of objects based on the evidence. This is further explained in Appendix C.

C. Update object-cluster connections

New detections are not only used to update existing objects but are also essential in linking objects to LiDAR clusters. These associations help maintain the connection between objects and clusters, enabling the system to track objects even when they are not visible to the camera.

To establish this object-cluster connection, the system checks which clusters could be part of the detection's bounding box. Similar to the object-association process, likelihoods are generated for associating detections with existing clusters. This is done using the positional likelihoods visualised in Figure 8. By transforming the positional uncertainties of clusters to the bounding-box frame of the detection, the Mahalanobis distance is used to determine the likelihood that a cluster is part of a detection. These likelihoods provide new evidence for the connection to the object associated with the detection.

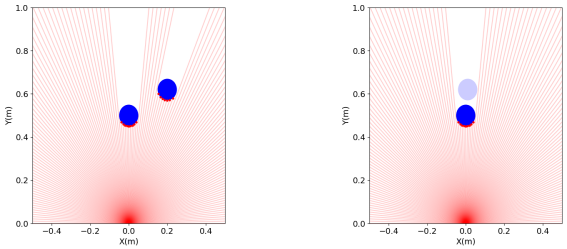
Similar to the type list of objects, each cluster has a list of possible parent objects with corresponding probabilities. Initially, the cluster is assigned no parent (-1) with a probability of 1.0. New potential parent objects are added and updated with their likelihoods, similar to Algorithm 2. Normalisation ensures that the total sum of probabilities remains 1.0. When the probability of an object surpasses a threshold of 0.5, it is designated as the parent of the cluster, and the object will follow the movement of the cluster. If multiple clusters are assigned to one object and they move in different directions (i.e. the distance between a cluster and the object increases), the likelihood that they are part of the same object decreases, as one or multiple of them are likely incorrectly assigned to that object.

If an object is removed while the clusters still exist, the probability that the cluster belongs to no object (-1) is increased by the probability that it was previously associated with the removed object.

VI. Human tracking

To track humans outside the camera's view, it is essential to accurately monitor the movement of their legs. However, as humans walk, one leg often moves behind the other in the LiDAR view, causing both legs to be clustered as a single entity. Since the legs alternate quickly between being clustered as one or two separate clusters, it becomes difficult to track them individually. This situation is visualised in Figure 9.

To address this, the cluster measurement association algorithm described in section V is modified. After all hypotheses are processed, the system checks any remaining unused hypotheses. If an existing cluster, which is part of a human (i.e., it has a human as its parent object), is not updated but has a hypothesis with a high probability, the system checks whether the measurement for that hypothesis has been used to update the other leg. If so, the measurement is used again to update the untracked cluster. This measurement is assumed to represent both legs, as they are very close together at that point. This ensures that both legs can still be tracked, even when one temporarily disappears.



(a) Human legs on LiDAR when the legs are separated. They appear as two clusters.

(b) Human legs on LiDAR when the they are close to each other. They appear as one cluster,

Fig. 9: Visualisation of a human's legs walking to the left in a LiDAR scan. When one leg is behind the other, it becomes invisible to the LiDAR. The human may appear as one or two clusters depending on the position of the legs during walking.

VII. Performance

Tests were conducted both in simulation and on the *Rober*.

A. Experiments in Simulation

A Gazebo simulation was used to evaluate the system's performance. The simulated environment replicates an indoor setting, including objects such as chairs and tables. Multiple actors follow predefined trajectories while others remain seated or stand still. The simulation is visualised in Figure 10.

In the simulation, the system was tested on three different aspects. First, the positional error of objects was measured using both camera-based and fusion-based estimation at various distances while the robot remained stationary. The objects tested included humans and chairs, as these are common in the robot's operating environment. Next, the system's tracking capability was evaluated by observing two humans walking around the robot. These individuals were tracked using LiDAR after the system associated them with human objects. Finally, a test was conducted with the

Fig. 10: Screenshot of the simulation environment, where multiple agents and objects are placed.

robot moving past objects. As the robot drove by, objects appeared and disappeared from the camera's view.

Object Positional Accuracy: To measure positional error, objects were placed in front of the robot, where they were visible to both the camera and the LiDAR. First, the depth estimation from the camera was compared to the actual distance, which is recorded as the camera error. Next, the object's position, once linked to the LiDAR clusters, was compared to its true position, recorded as the fusion error. These tests were conducted at distances of 3m and 6m from the robot, using humans and chairs as the test objects. The chairs were partially transparent, adding complexity to the camera's ability to estimate their position accurately. The tests are repeated three times and the average results are shown in Table I.

	Distance (m)			
	3		6	
Object type	Camera err.	Fusion err.	Camera err.	Fusion err.
Human	0.091	0.052	0.020	0.033
Chair	0.082	0.012	0.051	0.003

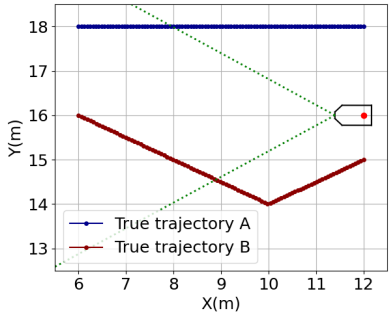
TABLE I: Average localisation error (in meters) of humans and chairs in simulation. Tests have been carried out with objects being 3m and 6m away from the center of the robot. The objects are far away from any walls or other objects in the background.

Additionally, these tests were repeated with the objects placed directly in front of a wall. Since the wall also appears as cluster(s) on the LiDAR, it could be mistakenly associated with the object, potentially introducing error. The results of these tests are shown in Table II.

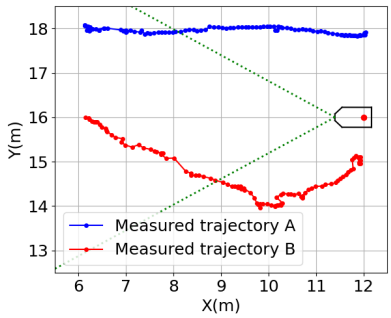
Tracking Accuracy: The robot must be capable of updating its environment in dynamic scenarios, particularly by tracking moving humans and other dynamic objects. To test this capability, two humans were positioned in the robot's camera view. While walking out of view, they were tracked using LiDAR. In Figure 11, the tracked and true positions of humans A and B are visualised. Human A walks in a straight line, while human B changes direction after moving out of the camera's view. Both humans were successfully tracked, with average tracking errors of 0.302m for human A and 0.367m for human B.

	Distance (m)			
	3		6	
Object type	Camera err.	Fusion err.	Camera err.	Fusion err.
Human	0.077	0.059	0.096	0.220
Chair	0.138	0.316	0.146	0.241

TABLE II: Average localisation error (in meters) of humans and chairs in simulation. Tests have been carried out with objects being 3m and 6m away from the center of the robot. The objects are placed close to a wall in the background



(a) True position of human A and human B walking past the robot.

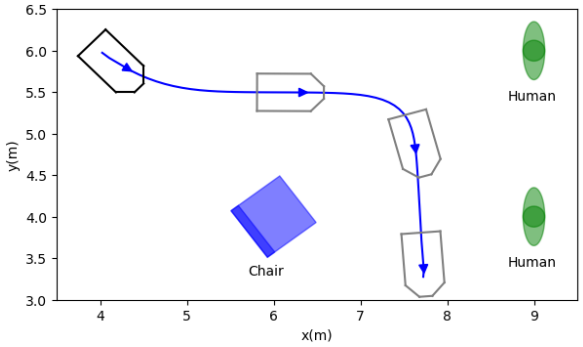


(b) Measured position of human A and human B walking past the robot.

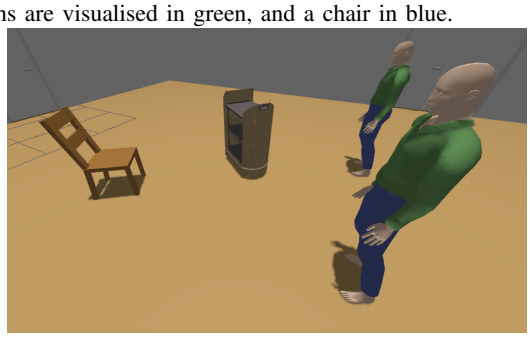
Fig. 11: True (a) and measured (b) position of human A and human B walking past the robot. Human A walks in a straight line, while human B takes a change in direction outside of camera view. The view of the camera is visualised with dotted green lines.

Dynamic Accuracy: Since the robot needs to be able to track objects while it is moving, the third testing area involves a moving robot in the environment. The robot follows a predefined path computed by the ROS2 navigation stack [27], and the positions of objects are tracked throughout the test. The driven path, which includes both straight segments and turns, is visualised in Figure 12a. The simulation environment and the resulting modeled environment are visualised in Figure 12b and Figure 12c, respectively. The

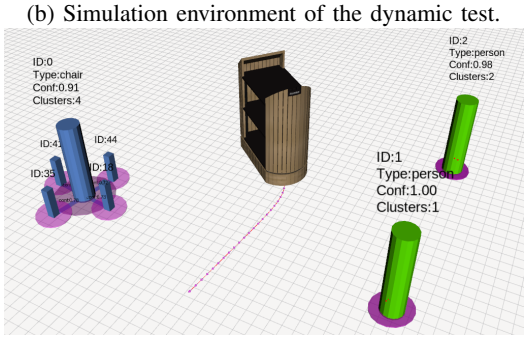
average position error of the objects was 0.044m for chairs and 0.074m for humans.



(a) Schematic of the driven path (blue line) by the robot. Two humans are visualised in green, and a chair in blue.



(b) Simulation environment of the dynamic test.



(c) Constructed environment by the system during the dynamic test.

Fig. 12: Overview of the dynamic test in simulation. The driven path is visualised in (a), with (b) giving a view of the simulation. Subfigure (c) visualises the output of the system, where the clusters are correctly linked to the corresponding objects.

B. *Experiments on the Robot*

In addition to the simulation tests, the system was evaluated on a real robot. Some hardware limitations, such as sensor delays, were observed. Despite this, the system successfully handled uncertainties and delays from various sensor sources by accurately linking LiDAR measurements to objects and tracking them within its environment. An example is visualised in Figure 1, where the robot was placed in a hallway containing static objects such as chairs and small potted plants. As shown in Figure 1a, the YOLO detection algorithm detected these objects with bounding boxes. Figure 1b visualises the resulting environment created by the system. Clusters visible on the LiDAR were correctly associated with the objects, with their confidence levels increasing over time as more data was gathered. Similarly, the system's confidence in the objects' classifications improved as additional evidence was collected. Similar to the tests in simulation, experiments performed on the robot focused on positional accuracy of objects, tracking of dynamic objects, and tracking the environment while in motion.

Object Positional Accuracy: The positional accuracy shown in tables I and II for the simulation was also tested on the real robot. A human and a chair were placed 3m and 6m in front of the robot, and the error in the camera depth estimation, as well as the positional error of the output of the system is computed. The tests have been done for objects placed in free space, and objects placed right in front of a wall. The results are shown in tables III and IV respectively.

	Distance (m)			
	3		6	
Object type	Camera err.	Fusion err.	Camera err.	Fusion err.
Human	0.125	0.002	0.903	0.004
Chair	0.472	0.014	1.283	0.081

TABLE III: Average localisation error (in meters) of humans and chairs in tests on the robot. Tests have been carried out with objects being 3m and 6m away from the center of the robot. The objects are far away from any walls or other objects in the background.

	Distance (m)			
	3		6	
Object type	Camera err.	Fusion err.	Camera err.	Fusion err.
Human	0.147	0.403	1.082	0.514
Chair	0.473	0.407	1.738	0.601

TABLE IV: Average localisation error (in meters) of humans and chairs in tests on the robot. Tests have been carried out with objects being 3m and 6m away from the center of the robot. The objects are placed close to a wall in the background

Tracking Accuracy: To test the system's tracking performance, the robot was positioned in an open space while a human walked from within the camera's view towards the side of the robot, eventually moving out of the camera's sight. The camera feed and the reconstructed environment are shown in Figure 14 in appendix A. Since no ground truth data was available for the human's trajectory, no specific metrics were computed. However, the system correctly tracked the human as they moved out of the camera's view and updated their position accordingly.

Dynamic Accuracy: In a similar dynamic test as in the simulation, the robot was tested while moving in an environment with surrounding objects. A chair and a human were placed in the environment, and the robot drove past them. The objects appeared and disappeared from the camera's view as the robot moved, and were tracked using LiDAR. Again, no ground truth data was available. However, the robot correctly associated the objects with clusters and was able to continue tracking them even when they were no longer visible to the camera. The camera feed with YOLO detections, as well as the system's output, is visualised in Figure 15 in appendix A.

VIII. Discussion and Future Work

Although the system can accurately recreate the environment around the robot, there are some limitations in the system that hinder further improvement of the accuracy.

Clustering Limitations: The LiDAR point cloud is first clustered before it is processed by the system. While this greatly simplifies entity tracking computations, it has some drawbacks. A larger object, such as a wall, is typically grouped as a single cluster. However, if the view is partially obstructed, the visible portions on either side may be incorrectly clustered as two separate entities. The robot itself has four blind spots, caused by structural poles that block visibility. This is illustrated in Figure 13, where the robot, positioned near a wall, sees the wall as two distinct entities in the clustered data. In reality, the robot should recognise this as a single object, with part of it occluded. This issue also arises when other objects obstruct the LiDAR's view. Furthermore, if the object blocking the LiDAR is moving, the 'shadow' that splits the background object will also move, making it seem as though the two clusters representing the background object are moving. This phenomenon occurs not only when another object is in motion but also when the robot itself moves, causing large background objects to appear as if they are shifting.

If clusters move within these blind spots, they can be tracked using the previously estimated velocity, but they are easily lost when changing direction or remaining in these spots for too long.

Object Position Uncertainty: Objects vary in shape and size, and from camera images alone, it is difficult to accurately estimate an object's length. This information can be crucial for linking objects with clusters. The uncertainty in the camera data dictates which clusters are associated with an object. Increasing the depth uncertainty allows clusters farther away to be linked to the object. This can be useful for objects like tables, where the legs are spread apart. However, this is not applicable for humans. The difference in fusion errors between Table I and Table II highlights this issue, as fusion errors for objects near walls increase due to wall clusters being incorrectly associated with objects. Solving this problem is challenging, as for objects like tables, it makes sense for clusters slightly behind the actual measurement to belong to the object, but for humans it is unlikely that their legs are far apart. Incorporating prior knowledge about object types and sizes could enhance system performance.

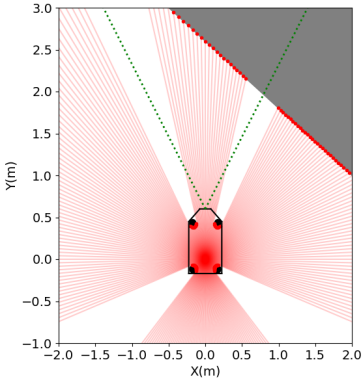


Fig. 13: LiDAR rays (red) from the robot when a wall (gray) is in view. The algorithm clusters this wall into two separate entities, despite it being a single object. The field of view of the camera is visualised with green dotted lines.

Furthermore, the camera's depth point cloud is inaccurate, causing depth points to appear farther away than they actually are. This means that, although the correct depth points are used to compute an object's position, it will still be placed slightly behind the actual object. Increasing the uncertainty in detection could help ensure that the correct clusters are more likely to be linked to the object. However, this approach also risks linking incorrect clusters from background objects, which is also evident from difference in the tables Table III and Table IV. The inaccuracy in the depth camera is most likely due to imperfect stereo calibration, and this could be improved through re-calibration.

Stationary Inference Issue: The system uses Bayesian inference to update its knowlegde of the environment. As sensors continuously gather evidence, the system becomes more confident about object classifications or object-cluster connections. However, when the robot is stationary, it repeatedly processes the same evidence. This issue is particularly noticeable for objects near walls, where the connection probability between the object and nearby clusters continues to increase. While clusters actually belonging to the object increase their connection probabilities faster, over time, clusters associated with the wall also reach high connection probabilities to the object due to constant evidence accumulation.

To address this, the weight of new evidence should decrease when the environment remains unchanged. Evidence from newly observed objects should carry more importance than evidence accumulated over long periods.

Ambiguous Object Detections: In some cases, the YOLO algorithm detects multiple objects at the same location. For instance, a human carrying a backpack may be

detected as two separate objects occupying approximately the same position on the map. Clusters may be linked to both the human and the backpack, with their probabilities fluctuating around 50%. When the human carrying the backpack moves out of the camera's view, only one of the objects is tracked, depending on which has the higher probability at the time. Incorporating prior knowledge—such as recognising that humans can carry other objects—could help resolve these ambiguities.

Occlusion Assumptions: As explained in subsection V-A, positional uncertainty is added to clusters at every update step. In the absence of new data, this leads to a uniformly expanding uncertainty regarding the cluster's location. While this is useful when an object reappears slightly displaced, the uncertainty can be improved by leveraging more knowledge from the system. To determine whether a cluster is occluded, the system compares the distance of the nearest three LiDAR rays to the cluster's distance in the model. If the cluster's distance is significantly shorter and remains unupdated, the system assumes the cluster is occluded, and its positional uncertainty grows uniformly. However, the cluster could be occluded by a narrow object, such as one of the poles in the robot's structure, which the LiDAR may see around. In this case, the uniform expansion of positional uncertainty is not optimal. Instead, there are areas where we can confidently say the cluster is not located. Narrowing the possible position of the occluded cluster could reduce mismatches with new data.

Object Type Similarity: As explained in section V, the association of YOLO detections with existing objects is based on both positional likelihood and type likelihood, considering the type and confidence levels of the YOLO detection and the existing object. However, some objects recognised by YOLO are very similar. For example, a large chair might be classified as either a chair or a couch. In contrast, the algorithm will not confuse humans with chairs. If the algorithm detects a human where a chair was previously observed, it is unlikely to be the same object. However, if it detects a couch where a chair was seen before, it is more likely to be the same object. This similarity could be leveraged to improve the accuracy of the model.

Fusion Localisation Uncertainty: As explained in section IV, both the LiDAR clusters and camera detections have a position uncertainty, consisting of both the sensor uncertainty and the localisation uncertainty. This information is useful for updating the positions of entities in the global map. However, when linking LiDAR clusters to detections, the localisation uncertainty is not required. Currently, the global positions of LiDAR clusters in the map are compared to the global positions of detections, as the new measurements from both sensors are implemented separately. However, since these measurements are taken at approximately the same time, localisation uncertainty becomes irrelevant, as both sensors would experience the same uncertainty. To improve this process, only the sensor uncertainties should be considered during the matching phase.

IX. Conclusion

In this paper, a stochastic 2D LiDAR-camera sensor fusion system for real-time dynamic object mapping is proposed. The architecture leverages the spatial resolution of the LiDAR and the object detection capabilities of the camera. Confidence in the existence and type of objects, as well as the probability that a tracked LiDAR cluster corresponds to an object detected by the camera, is continuously updated using Bayesian inference. Additionally, localisation and sensor uncertainties are accounted for by modeling measurements and environmental entities using Gaussian distributions. The fusion performance, applied with Bayesian update rules, has shown satisfactory results in both simulation and real life experiments, demonstrating an average static positional error of objects of 0.12m in simulation, and 0.25m across different experiments on the robot. Furthermore, the system performed well in tracking dynamic humans, with an average tracking error of 0.33m in simulation.

References

[1] T. Dalhuisen and B. Dalhuisen. (2024) Dalco robotics. [Online]. Available: https://dalcorobotics.nl/

[2] J. Dong, X. Fei, and S. Soatto, "Visual-inertial-semantic scene representation for 3d object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3567–3577.

[3] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[4] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," *CoRR*, vol. abs/1910.02490, 2019. [Online]. Available: http://arxiv.org/abs/1910.02490

[5] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," *CoRR*, vol. abs/1706.06629, 2017. [Online]. Available: http://arxiv.org/abs/1706.06629

[6] ——, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," *CoRR*, vol. abs/1804.09194, 2018. [Online]. Available: http://arxiv.org/abs/1804.09194

[7] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1352–1359.

[8] S. Yang and S. A. Scherer, "Cubeslam: Monocular 3d object detection and SLAM without prior models," *CoRR*, vol. abs/1806.00557, 2018. [Online]. Available: http://arxiv.org/abs/1806.00557

[9] B. Bescós, C. Campos, J. D. Tardós, and J. Neira, "Dynaslam II: tightly-coupled multi-object tracking and SLAM," *CoRR*, vol. abs/2010.07820, 2020. [Online]. Available: https://arxiv.org/abs/2010.07820

[10] J. Zhang, M. Henein, R. E. Mahony, and V. Ila, "VDO-SLAM: A visual dynamic object-aware SLAM system," *CoRR*, vol. abs/2005.11052, 2020. [Online]. Available: https://arxiv.org/abs/2005.11052

[11] Y. D. Kwon and J. Lee, "A stochastic map building method for mobile robot using 2-d laser range finder," *Autonomous Robots*, vol. 7, 1999.

[12] R. Soitinaho, M. Moll, and T. Oksanen, "2d lidar based object detection and tracking on a moving vehicle," *IFAC-PapersOnLine*, vol. 55, no. 32, pp. 66–71, 2022, 7th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322027495

[13] A. Ravankar, A. A. Ravankar, A. Rawankar, and Y. Hoshino, "Autonomous and safe navigation of mobile robots in vineyard with smooth collision avoidance," *Agriculture*, vol. 11, no. 10, p. 954, 2021.

[14] L. Fagundes-Junior, A. Gomes Caldeira, M. Quemelli, F. Martins, and A. S. Brandão, "Object detection in robot autonomous navigation using 2d lidar data: An analytical approach," 01 2024.

[15] K. Konstantinidis, M. Alirezaei, and S. Grammatico, "Development of a detection and tracking of moving vehicles system for 2d lidar sensors," *Delft Universitu of Technology*, 2020. [Online]. Available: https://resolver.tudelft.nl/uuid:103fe186-925e-46f7-8275-d746e7c47600

[16] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2d lidar," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1039–1063, 2015. [Online]. Available: https://doi.org/10.1177/0278364914562237

[17] J. Lee, M. Kim, , and H. Kim, "Camera and lidar sensor fusion for improving object detection," 2019.

[18] C. Wang, C. Ma, M. Zhu, and X. Yang, "Pointaugmenting: Cross-modal augmentation for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 11 794–11 803.

[19] T. Yin, X. Zhou, and P. Krähenbühl, "Multimodal virtual point 3d detection," *CoRR*, vol. abs/2111.06881, 2021. [Online]. Available: https://arxiv.org/abs/2111.06881

[20] K. Huang and Q. Hao, "Joint Multi-Object Detection and Tracking with Camera-LiDAR Fusion for Autonomous Driving," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9 2021. [Online]. Available: https://doi.org/10.1109/iros51168.2021.9636311

[21] T. Huang, Z. Liu, X. Chen, and X. Bai, "Epnet: Enhancing point features with image semantics for 3d object detection," *CoRR*, vol. abs/2007.08856, 2020. [Online]. Available: https://arxiv.org/abs/2007.08856

[22] A. Mulyanto, R. I. Borman, P. Prasetyawana, and A. Sumarudin, "2d lidar and camera fusion for object detection and object distance measurement of adas using robotic operating system (ros)," *JOIV Int. J. Informatics Vis 4.4*, 2020.

[23] G. H. Hwang, S. W. Lee, and J. Jeon, "Ros2 implementation of object detection and distance estimation using camera and 2d lidar fusion in autonomous vehicle," in *2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE)*, 2024, pp. 1–5.

[24] "Euclidean Cluster Extraction — Point Cloud Library 0.0 documentation." [Online]. Available: https://pcl.readthedocs.io/projects/tutorials/en/master/cluster_extraction.html

[25] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," *KI - Künstliche Intelligenz*, 2009. [Online]. Available: https://mediatum.ub.tum.de/doc/800632/941254.pdf

[26] slamtec, "RPLIDAR S2 Key parameters," 2024. [Online]. Available: https://www.slamtec.com/en/S2/Spec

[27] "Nav2 — Nav2 1.0.0 documentation." [Online]. Available: https://docs.nav2.org/

[28] Defence Science and Technology Laboratory (Dstl), "6 - Data association - multi-target tracking tutorial," 2024. [Online]. Available: https://stonesoup.readthedocs.io/en/latest/auto_tutorials/06_DataAssociation-MultiTargetTutorial.html

[29] G. M. Mimmack, S. J. Mason, and J. S. Galpin, "Choice of distance matrices in cluster analysis: Defining regions," *Journal of climate*, vol. 14, no. 12, pp. 2790–2797, 2001.

[30] J. K. Uhlmann, "Covariance consistency methods for fault-tolerant distributed data fusion," *Information Fusion*, vol. 4, no. 3, pp. 201–215, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253503000368

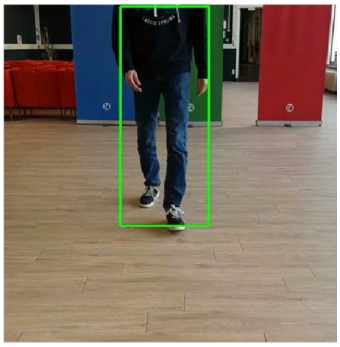
[31] B. D. Anderson and J. B. Moore, *Optimal filtering*. Prentice-Hall, Inc., 1979.

[32] luxonis, "YoloSpatialDetectionNetwork," 2024. [Online]. Available: https://docs.luxonis.com/software/depthai-components/nodes/yolo_spatial_detection_network/

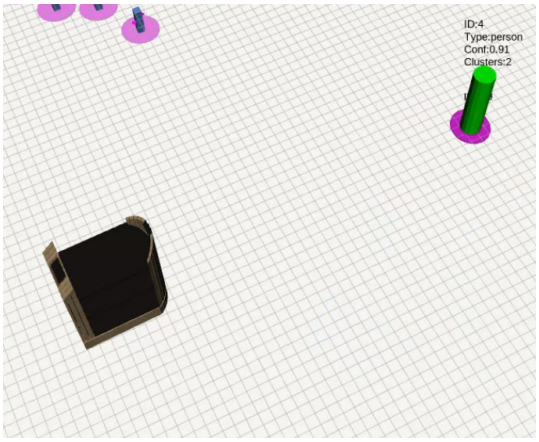
[33] E. Britannica, "Bayesian analysis — Probability Theory, Statistical Inference," 8 2024. [Online]. Available: https://www.britannica.com/science/Bayesian-analysis

APPENDIX

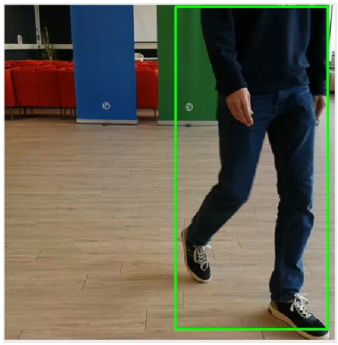
A. Test Figures



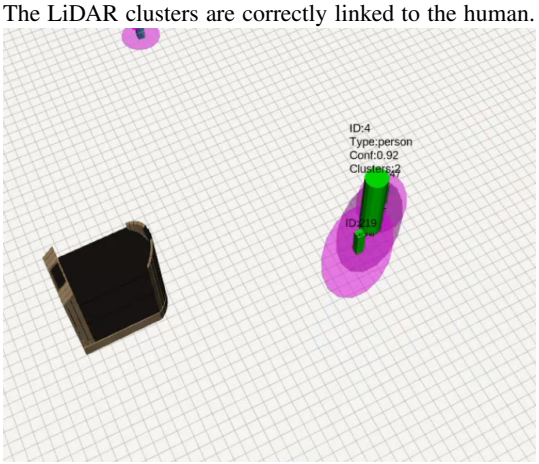
(a) Camera image with YOLO detections when human is in view.



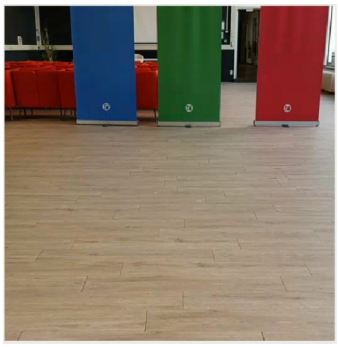
(b) Constructed environment when human is in view. The LiDAR clusters are correctly linked to the human.



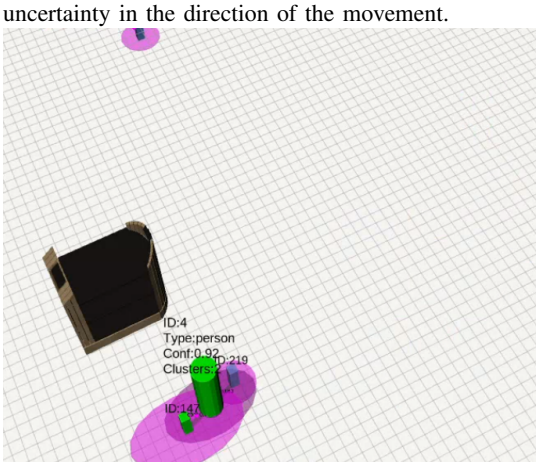
(c) Camera image with YOLO detections when human walks out of view.



(d) Constructed environment when human walks out of view. The human is tracked based on the clusters. As the human is moving, there is a higher positional uncertainty in the direction of the movement.

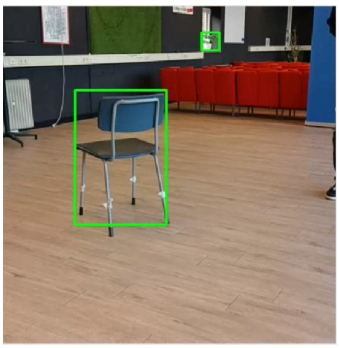


(e) Camera image with YOLO detections when human is not in view.

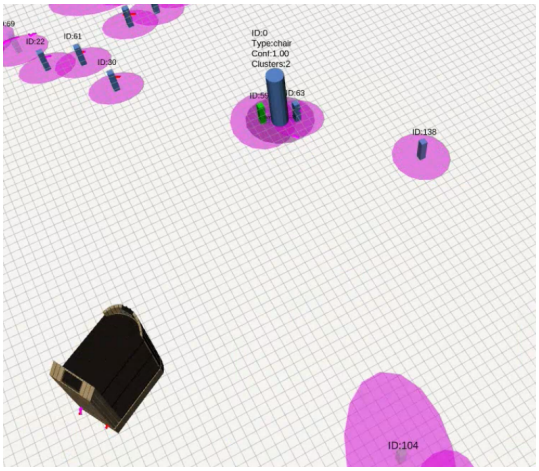


(f) Constructed environment when human is standing next to the robot, outside of camera view. It is still classified as a human and movements of the legs are tracked by the LiDAR.

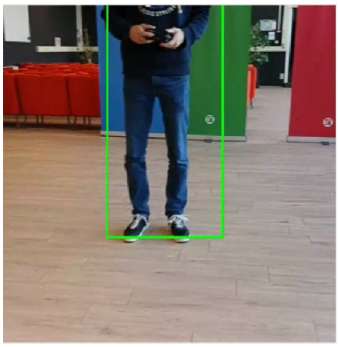
Fig. 14: Experiment with a dynamic object. The human starts in the view of the robot, and walks out of view. The robot is able to track the human with the associated LiDAR clusters and updates its position accordingly. Figures (a), (c) and (e) represent the camera image with the YOLO detections at different time stamps, while the figures (b), (d) and (f) are the corresponding constructed environments.



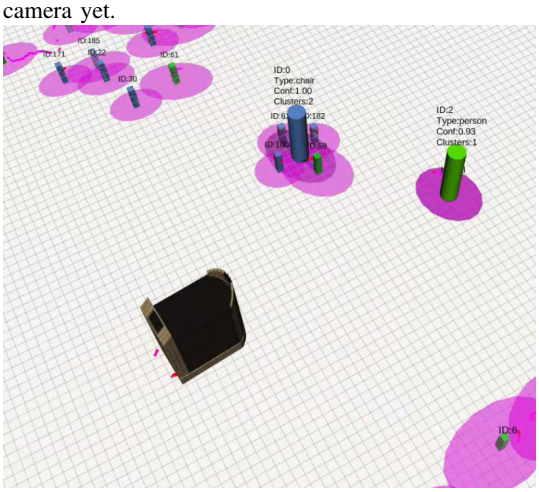
(a) Camera image with YOLO detections when chair is in view.



(b) Constructed environment when chair is in view. The visible LiDAR clusters are correctly linked to the chair. The human is seen on the LiDAR, but not on the camera yet.

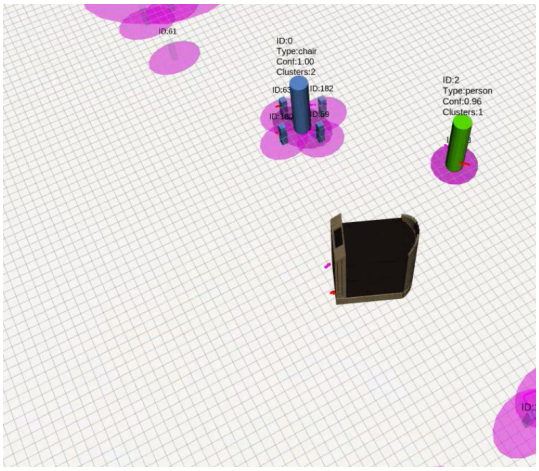


(c) Camera image with YOLO detections when chair moves out of view due to movement of the robot. The human is now detected by the YOLO algorithm.



(d) Constructed environment when the chair moves out of view due to movement of the robot. The chair is tracked based on the associated clusters. The human is correctly classified and linked to the cluster of the legs.

(e) Camera image with YOLO detections when the chair and the human are not in view anymore due to movement of the robot.



(f) Constructed environment when the chair and human are out of view due to movement of the robot. The robot still knows where the objects are.

Fig. 15: Experiment while driving with the robot. During the test, the robot sees a human and a chair which appear and disappear from the camera image due to the movement of the robot. Figures (a), (c) and (e) represent the camera image with the YOLO detections at different time stamps, while the figures (b), (d) and (f) are the corresponding constructed environments.

B. Camera Depth Estimation

When objects are detected by the camera, depth estimation can be performed using the stereo point cloud generated by the camera. The challenge lies in determining which points in the point cloud correspond to the object. Since the detections are represented as bounding boxes rather than pixel-level segmentations, a significant amount of background may be included in the detection. This is illustrated in Figure 16, where a chair is placed 1.5m in front of a wall.

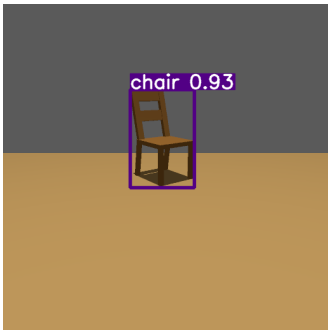


Fig. 16: YOLO detection of a chair placed in front of a wall.

By default, the Luxonis camera software estimates depth by taking a margin from the bounding box and computing the average distance of all points within that area [32]. While this method works well for solid objects, it can be problematic for objects such as chairs and tables, where a large portion of the bounding box may include background points, leading to depth estimates that place the object behind its actual position. In Figure 17, the density of the pointcloud is visualised. The first peak corresponds to the object, while the second peak represents the wall in the background. The green line indicates the true centroid of the object, while the red line represents the average depth of all points, which falls between the object and the wall, resulting in an error of 0.74m in this example.

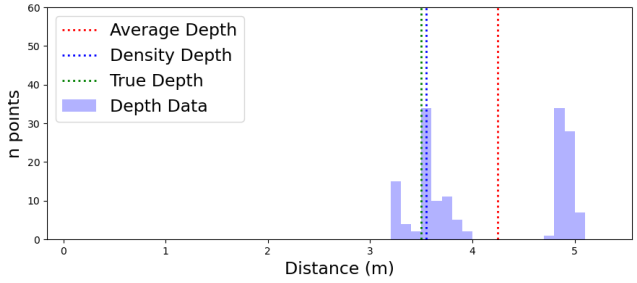


Fig. 17: Depth points of the camera within the bounding box. The red dotted line represents the average depth of all points, while the blue dotted line shows the depth computed by analyzing the point cloud density. The green line indicates the true center of the object.

To account for this issue, an alternative approach is used. Instead of taking the average of all the points, the point density is analysed. After applying a moving average filter, the first real peak in the depth density above a certain threshold is taken as the distance to the object. This is shown by the blue dotted line in Figure 17, reducing the error to 0.05m. This method significantly reduces the uncertainty in the depth estimation of the object, bringing the detection closer to the actual position of the object.

C. Existence Probability Updates

As a cluster is updated, confidence in its existence should increase. Bayesian updates are applied, where the probability that a hypothesis H (that a cluster exists) is true, given evidence E, is computed as [33]:

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)} \tag{25}$$

where $P(H \mid E)$ is the posterior probability, which is the updated probability based on the new evidence. $P(E \mid H)$ is the likelihood, the new evidence provided by the sensors. $P(H)$ is the prior probability and $P(E)$ the marginal likelihood. The marginal likelihood represents the general likelihood of observing the evidence E. It can be expanded to [33]:

$$P(E) = P(E \mid H)P(H) + P(E \mid \neg H)P(\neg H) \tag{26}$$

where $\neg H$ is the logical negation of H (i.e. "not H"). The probability $P(\neg H)$, the likelihood that H is false, is calculated as $P(\neg H) = 1 - P(H)$ since the sum of the probabilities of H being true and false must equal 1. $P(E \mid \neg H)$ represents the likelihood of observing evidence E when hypothesis H is false. This value typically needs to be estimated, and in this case, it has been set to 0.1. To filter out minor fluctuations, the updated probability is calculated as a weighted average of the prior and the posterior, with the prior weight set at 0.7 and the posterior weight at 0.3.

Unlike object type updates, where the YOLO detection algorithm provides the new evidence, the existence probability must be estimated. When a cluster is updated, a likelihood of 0.99 is used. However, in certain cases, the belief in the existence of a cluster should decrease. These likelihood values determine how quickly the existence probability of a cluster diminishes under different conditions:

- Occlusion of static clusters; When the cluster is blocked by another object or wall, preventing updates, it is assumed the object is still there but unverified. The likelihood of existence is estimated at 0.07.
- Occlusion of dynamic clusters: Similar to static clusters, there is no data to update the cluster. However, since dynamic clusters are unlikely to remain in the same position or maintain the same velocity, the likelihood of existence is estimated at 0.02.

- Disappeared clusters: If the cluster should be visible but is not updated with new data, it is likely that it has disappeared from the area. In such cases, the likelihood of existence is estimated at 0.0001, indicating a rapid decrease in confidence.

The outcome of these scenarios is visualised in Figure 18.

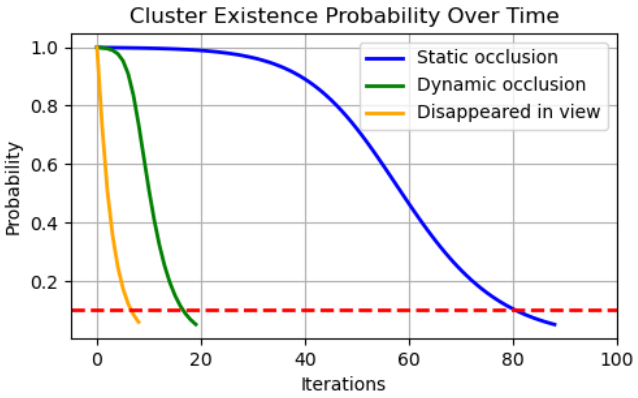


Fig. 18: Existence probability in situations where a cluster is occluded (static or dynamic) or when an object has disappeared from view. The y-axis represents the existence probability of the clusters, initialized at 1.0 for clarity and updated using Bayesian inference. The x-axis shows the number of update iterations. The system operates at 10 Hz.

A similar approach is used for determining the existence probability of objects. When an object is detected in multiple frames, confidence in its existence increases with a likelihood equal to the YOLO detection confidence. If an object is expected to be in view but is not updated, its existence probability rapidly decreases, with a likelihood of 0.001. Additionally, when objects are out of view, their existence probability is based on the existence probability of their associated clusters. Objects without clusters are assigned an existence likelihood of 0.07 when out of view, as their position or existence cannot be updated by either sensor.