Semantic Annotation of Lidar Data Using RGB Cameras MiRaculously Working

Adelina Derihvist
adder24@student.sdu.dk

Alberto Vicente Chacón
alvic24@student.sdu.dk

Tabea Sudermann
tasud24@student.sdu.dk

***Abstract*—Standard 2D maps used in mobile robotics lack semantic information, limiting the robot's ability to distinguish between static and dynamic obstacles. This work utilizes manual calibration methods and image segmentation models to semantically annotate LiDAR point clouds using a "Point Painting" technique. A 2D single plane LiDAR sensor and an RGB camera are used to collect the necessary data.**

We conducted a comparative study between the state-of-the-art real-time detector YOLOv11 and the zero-shot model SAM 3, using the COCO128-seg dataset. Experimental results demonstrate that SAM 3 significantly outperforms YOLOv11 in segmentation fidelity, achieving a mean IoU of 0.65 compared to 0.50 (+30% relative improvement). However, this comes at a computational cost: SAM 3 operates at $\sim$1.3 FPS compared to YOLOv11's $\sim$24 FPS. The proposed system offers a robust solution for high-fidelity semantic mapping missions where precision is prioritized over traversal speed.

***Index Terms*—Semantic Segmentation, 2D LiDAR, Calibration, Sensor Fusion**

I. Description of the industrial challenge

The mobile robot MiR is largely used by different customers that are striving to take a step forward towards automation. These mobile robots provide extra help in dealing with repetitive tasks; they allow medical staff to focus on more important work than transporting an item from A to B. If looking at real-world cases, mobile robots are used for transporting medicine [1] or blood samples [2]. Mobile robots are also used in warehouses or retail stores as they can navigate autonomously and help automate repetitive processes [3].

The robot navigates in a busy environment such as the hospital through a map that is being created before starting the automation process. The mapping is done through a Light Detection and Ranging (LiDAR) sensor scan, that gives "knowledge" to the robot about its environment. The map is always used to define the target location for each task.

Since the sensor scanning does not provide information about what type of object it is detecting, it is necessary to "clean" the map manually from noisy readings or from irrelevant objects, like people or other moving objects that do not represent a permanent obstacle for the robot. This task is necessary as the robot does not update its map when sent on a mission; for an updated map, a new scanning process needs to be done. Therefore, this project is concerned with adding a semantic label to each point of the map through RGB image segmentation and classification into three categories: static, semi-static and dynamic. This solution can contribute to an updated map in future development. This report aims to provide an integrated solution for semantic annotation of LiDAR readings based on a combination between the semantic information from an RGB camera and the depth information from LiDAR.

II. State of the art

In the context of semantic segmentation for autonomous driving (e.g., SemanticKITTI [4]), complex deep learning-based fusion strategies are often computationally prohibitive. These methods typically process sensor streams independently before combining features [5], which significantly increases model complexity and delays sensor readings [6].

A more efficient alternative is to project the 2D LiDAR data directly onto the RGB-D camera image plane. This allows for direct spatial correlation and semantic labeling of the LiDAR points, avoiding the overhead of multi-stream neural networks. This projection technique is well-suited for real-time systems and aligns with practical implementations such as that of Benayed et al. [7], where they propose a framework for multi-modal object detection. Their solution aims for low-cost sensor fusion for a semantic and geometrical perception of the environment.

The state-of-the-art research shows projects that use a point painting method for transferring the information from the RGB data to LiDAR data. Specifically, in the PointPainting paper [8] three different methods were investigated for their semantic segmentation accuracy and the final point projection solution was tested on the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago (KITTI) and nuScenes datasets, where the implemented solution has shown more robust and accurate performance.

A more recent implementation, SPNet [9], utilizes a cross-modal knowledge distillation framework, a concept originally introduced by Gupta et al. [10]. In this setup, a 'teacher' network is trained on LiDAR data enriched with RGB semantics (via point-painting), while a 'student' network (sharing the same architecture) operates on raw LiDAR data.

The goal is to transfer the teacher's privileged semantic understanding to the student. By using the teacher's output as an auxiliary supervisory signal, the student learns to identify semantic features within the sparse point cloud that are

otherwise difficult to detect, resulting in more robust 3D object detection on the KITTI dataset.

In [11] the sparsity readings of the LiDAR sensor are addressed and the proposed method fuses the point clouds with the camera, making use of the dense texture information given by images. The network architecture includes an extraction of the region of interest before the encoding layer, which has shown improved accuracy of object detection when compared to the LiDAR two-stage detector, when tested on overlapping views of objects in the image.

In the domain of real-time object detection, the You Only Look Once (YOLO) series remains the industry standard for high-speed applications [12]. Unlike two-stage detectors that rely on region proposals, YOLO treats object detection as a single regression problem, predicting bounding boxes and class probabilities directly from full images in one evaluation. Recent iteration, such as YOLOv11, has optimized the trade-off between latency and accuracy, achieving inference times as low as 2-5ms on local devices [13]. This makes the architecture highly suitable for industrial environments where low latency is critical for safety and process control.

Conversely, the advent of Foundation Models has introduced the Segment Anything Model (SAM), with the latest iteration, SAM 3, released by Meta in late 2025 [14]. While previous versions focused on prompting single objects, SAM 3 introduces "Promptable Concept Segmentation," allowing the model to identify and segment every instance of a semantic concept (e.g., "all pallets" or "obstacles") across an image or video sequence without specific training on those classes (zero-shot capability). Although SAM 3 offers superior generalization and open-vocabulary understanding compared to the closed-set nature of YOLO, it is computationally heavier, often requiring server-grade GPUs for real-time performance.

The fusion of 2D LiDAR and RGB-D camera images combines the strengths of both: The RGB-D camera provides rich semantic information for detection and classification, while the 2D LiDAR provides precise spatial information. Together, they can be used to continuously update the map based on semantic segmentation and depth information.

While there is more research on fusing 3D LiDAR with RGB-D cameras, 2D LiDAR offers a cheaper and less computationally expensive solution. However, a single-line 2D LiDAR scanner, like the one integrated in the MiR robot, only captures points in a single horizontal plane. This makes it more difficult to align the LiDAR line with the corresponding pixels in the camera image. The calibration process described in the public repository [15] involves manually selecting descriptive feature points both in the LiDAR and camera readings that can be matched afterwards. From multiple data collection sequences, these points are refined to minimize the reprojection error between the sensor frames.

For an effective fusion, an intrinsic and extrinsic calibration of the sensors is required: Intrinsic calibration to adjust for lens distortions and extrinsic calibration to determine the relative pose between the LiDAR and camera coordinate frames.

Based on the analysis of the state of the art, this project aims to address the industrial challenge by implementing a projection-based fusion system. A critical decision in this pipeline is the choice of the semantic segmentation engine. We have decided to perform a comparative analysis between YOLOv11, representing the state-of-the-art in supervised, real-time efficiency, and SAM 3, representing the cutting edge of zero-shot, open-vocabulary segmentation. This comparison will evaluate which model offers the optimal balance between inference speed and segmentation robustness for our specific industrial environment.

III. Approach

In this chapter, our approach for a sensor fusion framework in indoor environments is described, which is capable of semantically segmenting LiDAR points. First, we estimate the intrinsic parameters of the camera. Then we calibrate the LiDAR sensor and camera by estimating the extrinsic parameters to be able to project LiDAR points to corresponding pixels in the image. Afterwards, we describe how we used the YOLOv11 and SAM 3 models to semantically segment the images and lastly, how to label the corresponding LiDAR points.

A. Sensor Calibration

The core of our perception system relies on the fusion of the 3D point cloud of the LiDAR sensor with 2D RGB images. This fusion enables the projection of semantic labels from the image plane onto the 3D spatial data.

The calibration process aims to identify the extrinsic parameters, i.e. the rotation matrix R and translation vector t, which describe the rigid-body transformation between the LiDAR and camera coordinate frames. This requires first to perform an intrinsic camera calibration.

The camera intrinsics describe the internal parameters of the camera, e.g. the focal lengths, and define how 3D points in the camera coordinate frame are projected onto the 2D image plane. Accurate intrinsic parameters are essential for our task, as errors directly affect the spatial alignment between image pixels and corresponding 3D points.

The camera's intrinsic parameters were estimated using the OpenCV library [16] with an ARUCO 4x4 checkerboard pattern. The resulting intrinsic matrix K expressed in pixels is:

$$K = \begin{bmatrix} 601.80 & 0 & 328.17 \\ 0 & 600.12 & 240.45 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

We performed the extrinsic calibration using the method proposed by Hong et al. [15], which estimates the transformation matrix that maps points from the camera frame to the LiDAR frame ($T_{C \to L}$). We recorded 30 pairs of 2D LiDAR scans and camera images which included an object that was detectable in both LiDAR data and image, with samples from different perspectives and distances. After finding correspondences in LiDAR data and the images, a Perspective-n-Point (PnP) algorithm was used to find the rotation $R_{C \to L}$ and

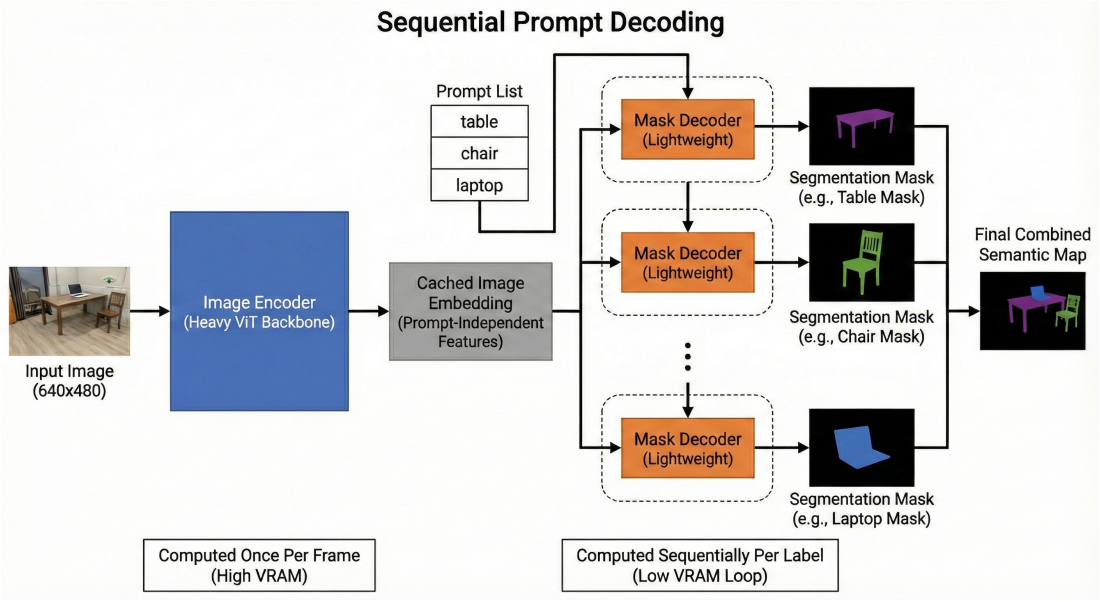

Fig. 1: Diagram of SAM3 model architecture

translation $t_{C\to L}$ such that the reprojection error is minimized. The resulting extrinsic calibration yielded a root mean square (RMS) error of 5.158 pixels between the projected 3D points and the actual detected points in the image over the 30 samples. The identified parameters from camera to LiDAR are:

$$T_{C\to L} = \begin{bmatrix} R_{C\to L} & t_{C\to L} \\ 0 & 1 \end{bmatrix} \tag{2}$$

$$R_{C\to L} = \begin{bmatrix} -0.7135 & -0.7005 & -0.0152 \\ -0.0827 & 0.1056 & -0.9910 \\ 0.6958 & -0.7057 & -0.1333 \end{bmatrix} \tag{3}$$

$$t_{C\to L} = \begin{bmatrix} -0.2048 \\ 0.0836 \\ -0.0447 \end{bmatrix} \text{ m} \tag{4}$$

To project a 3D LiDAR point onto the 2D image plane, the transformation matrix $T_{L\to C}$ from LiDAR to camera frame is required. It is derived by the inverse of $T_{C\to L}$ with the following equations:

$$\begin{aligned} T_{L\to C} &= T_{C\to L}^{-1} \\ T_{C\to L}^{-1} &= \begin{bmatrix} R_{C\to L}^{-1} & -R_{C\to L}^{-1} t_{C\to L} \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{5}$$

The projection of a LiDAR point $P_L = [x, y, z, 1]^T$ into the image frame is then given by:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_C = T_{L\to C} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_L \tag{6}$$

The RMS error of 5.158 pixels demonstrates a high degree of geometric consistency between the LiDAR and camera frames. This error is well below typical thresholds reported for sparse feature tracking (e.g., visual SLAM). Our system operates in indoor environments where target objects (e.g., furniture, humans) occupy significant spatial regions within the image frame. Given that the task involves associating LiDAR points with dense semantic masks, rather than pixel-perfect feature matching, this calibration margin allows for robust point-to-mask association without misclassifying objects. The error magnitude is significantly smaller than the average bounding box of the semantic classes of interest at the robot's operational range.

B. Resource-Aware Semantic Segmentation

To classify objects within the scene, we employ a dual-model strategy targeting indoor environments. We evaluate two distinct architectures: YOLOv11 for real-time object detection, and MetaAI's Segment Anything Model 3 (SAM 3) for high-fidelity, open-vocabulary segmentation.

The original YOLOv11 model is able to identify 79 different objects, many of which are irrelevant for indoor robotic applications. Therefore, the model was initially retrained using the HomeObjects-3K dataset from ultralitycs [17] to refine it for our use case. An example of images that this dataset offers can be seen in Figure 2.

The training consisted of 100 epochs with 2285 images. These images, which include indoor scenes, are segmented for 12 different classes. The results of the training are represented in Figure 4 with the F1-score for each class. The F1 score for all the classes is highest at confidence 0.519 with a score of 0.69. The F1-score is a metric used for evaluating the precision of the model and is given by:

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{7}$$

Fig. 2: HomeObjects-3K Dataset Examples

Where TP is the true positive prediction, FP the false positive and FN the false negative.

As this dataset does not include annotations for people, the model cannot reliably detect humans. Therefore, the original model is used for further object detection and point painting.

A significant challenge in deploying large foundation models like SAM 3 on mobile robot hardware is the high memory needs, which can break the inference pipeline. During the experiments, a Laptop RTX 4070 was used, which only has 8GB of VRAM. To address this, we used the following inference architecture:

- **Process Separation:** We isolate the camera acquisition task and the CUDA inference engine into separate operating system processes. This eliminates segmentation faults during continuous operation.
- **Adaptive Resolution Scaling:** To prevent Out-Of-Memory (OOM) errors, raw images (640×480) are downscaled ($w = 480$) for the vision encoder. Deep learning models consume GPU memory in two forms: static weights (constant) and dynamic activations (intermediate feature maps). Since activation memory scales with the total pixel count ($Area = W \times H$), a linear reduction of 25% in width results in a quadratic reduction in total surface area ($\lambda^2 \approx 0.56$). This allows a memory saving of approximately 44% on activation tensors, allowing the Transformer backbone to fit within the 8GB VRAM budget.
- **Sequential Prompt Decoding:** The SAM 3 architecture has an asymmetry between its components: a computationally intensive Vision Transformer (ViT) backbone (Image Encoder) and a lightweight Mask Decoder. The ViT processes raw pixel data to generate an image embedding, a step that consumes the majority of the VRAM. Crucially, this embedding is prompt-independent, meaning the visual features of the scene remain constant regardless of the semantic query. Taking advantage of this, we implement a caching strategy: the heavy ViT inference is executed only once per frame to produce the image embedding (see Figure 1). This cached embedding is then reused by the lightweight decoder, which is called for each class prompt (e.g., ”person”, then ”table”). This approach reduces the heavy encoding cost and ensures that peak memory usage corresponds to a single decoding task, decoupling VRAM consumption from the number of semantic categories queried.

This approach allows us to utilize the superior segmentation accuracy of SAM 3 while maintaining the robustness required for robotic mapping.

C. Semantic Labeling of LiDAR Points

Once the semantic masks are obtained for the images, we project the labels onto the 3D LiDAR points by using the estimated transformation matrix from LiDAR to camera frame. Only the LiDAR points which lay within the image frame are considered. For each LiDAR point the label of the corresponding image pixel is assigned to it. An example of the projected and annotated LiDAR points can be seen in Figure 3.

The final step involves categorizing the environment into three mobility classes: *Static*, *Semi-Static*, and *Dynamic*. The YOLO and SAM 3 models are configured to detect specific object classes (e.g., ”person”, ”chair”, ”table”). These can be systematically remapped to the defined classes:

- Dynamic: Objects that can move on their own (e.g., people, pets).
- Semi-Static: Objects that are movable but currently stationary (e.g., chairs, tables, laptops).
- Static: Any LiDAR point that projects into the image frame but does not intersect with a Dynamic or Semi-Static segmentation mask is classified as Static structure (e.g., walls, pillars, ceilings).

This logic enables the robot to build a semantic map that differentiates between permanent infrastructure and temporary obstacles.

IV. Experimental Prototype

The final integration of the project has been tested for accuracy and stability. As the fundamental part of the project is the segmentation of the video, this was tested first. Initial testing consisted in live visualization of both model performances and evaluating how well they can segment and detect objects in the scene. As there is no ground truth available for live testing, both models were tested on small data and compared the results with available ground truth. The dataset used for testing is the Common Objects in Context (COCO), specifically COCO128-seg, that contains the first 128 images from the entire data.

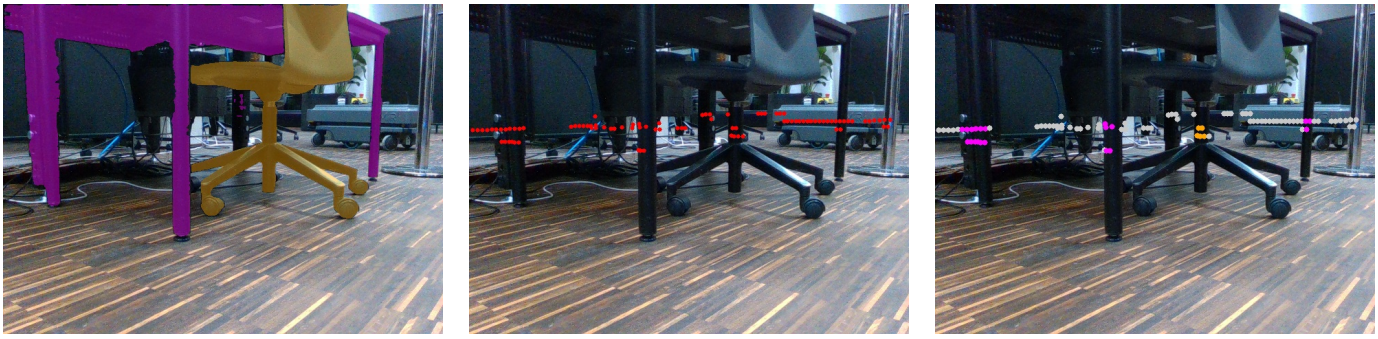

Fig. 3: Left: Semantic segmentation of objects in the image using SAM 3. Middle: LiDAR points projected into the image frame. Right: Labeled LiDAR points

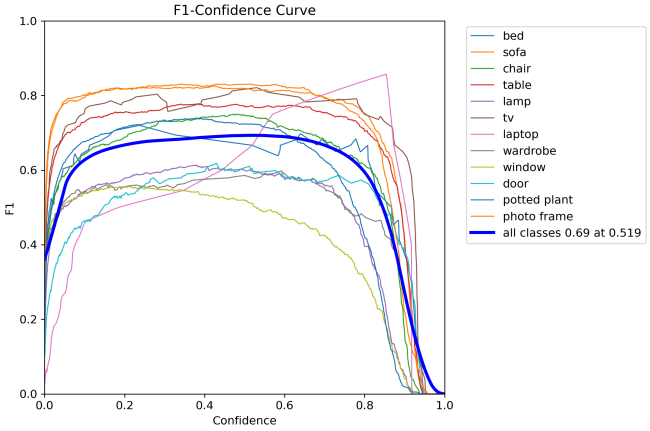

Fig. 4: F1 Confidence curve for YOLOv11 training

The data is structured such that every raw image has a respective text file with the corresponding masks. The mask is represented by the coordinate points of a polygon, outlining the shape of the object. Each row in the text file represents one object in the scene, and it starts with the representative class number. There is a total of 79 classes that include a variety of random objects from indoors and outdoors scenes.

The YOLOv11 and SAM 3 were given all 128 images to segment with the specified 79 classes. After computing the masks from both models they were compared to the ground truth mask by calculating the Intersection over Union (IoU) and Dice metrics.

The IoU is a common metric used for evaluation of segmentation tasks and is computed by the following formula:

$$IoU = \frac{A \cap B}{A \cup B} \tag{8}$$

where A is the area of the mask generated by the model and B is the area of the ground truth mask.

The Dice metric is given by:

$$Dice = \frac{2 * |A \cap B|}{|A| + |B|} \tag{9}$$

which is more sensitive to overlap between the two masks.

An experimental test was done for validating the reprojection of LiDAR points into the image. The experiment consisted in selecting a scene with specific features that helps identify if the reprojected points match the features in the scene. As there is no ground truth available for the reprojection other than the calculated RMS value after the calibration, this method was considered sufficient for a proof of concept prototype.

Since the reprojection of the LiDAR was tested live on real features from the scene, the final integration that consists of painting the LiDAR points according to the segmentation masks was also tested on live features.

The robot was moved around the lab to identify if the appearance of new objects will be detected correctly by the LiDAR. Additionally, it was observed that the points were assigned the correct color which corresponds to the color of the given mask. This experiment was done using the YOLOv11 model, as it is able to run on a video sequence and not just static images.

V. Results

The performance of the SAM 3 model in terms of IoU and Dice metrics has exceeded the one of the YOLOv11 model significantly. As detailed in Table I, SAM 3 achieved a mean IoU of 0.650 compared to YOLOv11's 0.503, representing a relative improvement of approximately 30% in segmentation fidelity.

However, this accuracy comes at a computational cost. While YOLOv11 operates comfortably in real-time at ∼24.2 FPS, the SAM 3 pipeline operates at ∼1.3 FPS. This confirms that while SAM 3 provides the pixel-perfect masks necessary for high-quality mapping, it is too slow for reactive obstacle avoidance, whereas YOLOv11 trades segmentation quality for high-speed processing.

TABLE I: Segmentation Performance Benchmark

Model	mIoU	mDice	Speed (FPS)
YOLOv11	0.503	0.571	∼ **24.2**
SAM 3	**0.650**	**0.732**	∼ 1.3

An example of the segmented images from real recordings can be seen in figure 5. It is quite visible in this example that the SAM 3 segmentation is more accurate.

Fig. 5: Segmentation masks: image to the left - SAM 3, image to the right - YOLOv11

During the final experiment where the robot was moving around the scene, an occlusion error was spotted due to the different field of views of the LiDAR and camera. Since the front LiDAR is placed in the corner of the robot and the camera is placed right in the middle of the front part, some points that are read from the background are reprojected into the foreground. As the system relies on segmentation for painting the LiDAR points, some of the background points are annotated as foreground objects. For example, in Figure 6, if a person is detected by both the LiDAR and the camera, there is a double line appearing in the pixels that represent a human in the image. This only occurs on the left side of the image, as the LiDAR is situated in the left corner of the robot.

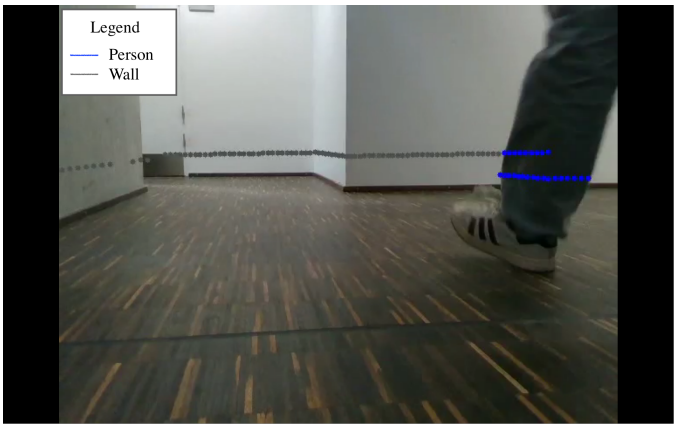

Fig. 6: LiDAR-Camera Projection Occlusion. Grey and blue points correspond to static and dynamic obstacles, respectively. The misalignment between the side-mounted LiDAR and central camera causes background points to incorrectly project onto foreground masks (visible as a "double line" on the right).

VI. Conclusion

The implemented solution works for creating an annotated 3D map, by coloring points according to their respective segmentation classes. The SAM 3 model has shown a better performance in comparison to YOLOv11, however it is important to mention that SAM 3 requires high computational power for live videos. The YOLOv11 model on the other hand, is lighter in terms of computation requirements but has a lower performance for segmentation. Given that the segmentation masks are used for a 2D map, this method is still suitable for real-time robotic applications.

Another limitation in this project is the erroneous annotation of the points reprojected due to occlusion. This causes some of the points that belong to different classes in the real scene to be considered as part of the same class when reprojecting. Even if the wrongly annotated points will be corrected once the robot moves from a different perspective, it is worth considering a further filtering process for these points. A filtering method can be implemented based on the assumption that any integral object can only have one line reprojected into the image. As this error only happens to the points in the background, the lower line of the two seen in the mask will be considered.

Lastly, but not least, the prototype system does not interfere with the robot map that is used for sending the robot to a certain mission. The next step in this project is to update the map and based on the nature of the object identified in the scene decide if it is required in the map or not. This solution, brings a more efficient route planning for the robot, as dynamic objects will not be registered in the map.

References

[1] G. A. Kebede, A. A. Gelaw, H. Andualem, and A. T. Hailu, "Review of the characteristics of mobile robots for health care application," *International Journal of Intelligent Robotics and Applications*, vol. 8, no. 2, pp. 480–502, 2024. [Online]. Available: https://doi.org/10.1007/s41315-024-00324-3

[2] Centre for Clinical Robotics (CCR), "Prøvsen (2022–2023)," https://ccrdenmark.com/robot-projects/provsen-2022-2023, Centre for Clinical Robotics, Odense University Hospital, 2023, accessed: 2025-12-15. [Online]. Available: https://ccrdenmark.com/robot-projects/provsen-2022-2023

[3] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020.
[4] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9297–9307.
[5] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.
[6] F. Pollach, T. Michalke, and C. Waldschmidt, "Low latency and low-level sensor fusion for automotive use-cases," in *2020 IEEE International Radar Conference (RADAR)*. IEEE, 2020, pp. 828–833.
[7] W. Benayed, I. Mabrouk, M. S. Masmoudi, and W. B. Abdelaziz, "Lidar 2d and camera fusion for adas: A practical approach with yolov9 and ros2 framework," *IEEE Access*, vol. 13, pp. 155 500–155 519, 2025.
[8] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4604–4612.
[9] B. Ju, Z. Zou, X. Ye, M. Jiang, X. Tan, E. Ding, and J. Wang, "Paint and distill: Boosting 3d object detection with semantic passing network," in *Proceedings of the 30th ACM International Conference on Multimedia*. ACM, pp. 6884–6893.
[10] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2827–2836.
[11] X. Xu, S. Dong, L. Ding, J. Wang, T. Xu, and J. Li, "ccccccfusionrcnn: Lidar-camera fusion for two-stage 3d object detection," *Remote Sensing*, vol. 14, no. 13, 2022.
[12] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 7464–7475.
[13] G. Jocher, A. Challis, and S. Qiu, "Yolov12: Redefining real-time object detection with latency-aware architecture," *arXiv preprint arXiv:2502.10987*, 2025, ultralytics.
[14] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Sam 3: Promptable concept segmentation at scale," *arXiv preprint arXiv:2510.04567*, 2025.
[15] E. Hong, "Camera 2d lidar calibration," https://github.com/ehong-tl/camera_2d_lidar_calibration, 2019, accessed: 2025-12-01.
[16] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120–125, 2000.
[17] G. Jocher and M. Rizwan, "Ultralytics datasets: Homeobjects-3k detection dataset," May 2025. [Online]. Available: https://docs.ultralytics.com/datasets/detect/homeobjects-3k/

Individual contribution:

Task Area	Adelina	Tabea	Alberto
Intrinsic Calibration	0%	30%	70%
Extrinsic Calibration	25%	50%	25%
Experiments	40%	30%	30%
YOLO	90%	5%	5%
SAM3	5%	5%	90%
Report Writing	40%	40%	20%

LLMs were used for debugging and coding assistance.