EasyChair Preprint

№ 10724

Artifacts Mapping: Multi-Modal Semantic Mapping for Object Detection and 3D Localization

Federico Rollo, Gennaro Raiola, Andrea Zunino, Nikolaos Tsagarakis and Arash Ajoudani

August 15, 2023

Artifacts Mapping: Multi-Modal Semantic Mapping for Object Detection and 3D Localization

Federico Rollo[†,‡,§], Gennaro Raiola[†,‡], Andrea Zunino[†,‡], Nikolaos Tsagarakis[‡], Arash Ajoudani[‡]

***Abstract*— Geometric navigation is nowadays a well-established field of robotics and the research focus is shifting towards higher-level scene understanding, such as Semantic Mapping. When a robot needs to interact with its environment, it must be able to comprehend the contextual information of its surroundings. This work focuses on classifying and localising objects within a map, which is under construction (SLAM) or already built. To further explore this direction, we propose a framework that can autonomously detect and localize predefined objects in a known environment using a multi-modal sensor fusion approach (combining RGB and depth data from an RGB-D camera and a lidar). The framework consists of three key elements: understanding the environment through RGB data, estimating depth through multi-modal sensor fusion, and managing artifacts (*i.e.*, filtering and stabilizing measurements). The experiments show that the proposed framework can accurately detect 98% of the objects in the real sample environment, without post-processing, while 85% and 80% of the objects were mapped using the single RGBD camera or RGB + lidar setup respectively. The comparison with single-sensor (camera or lidar) experiments is performed to show that sensor fusion allows the robot to accurately detect near and far obstacles, which would have been noisy or imprecise in a purely visual or laser-based approach.**

I. Introduction

To boost navigation autonomy and contextual awareness of mobile robots in unstructured environments, geometric information collected from the surroundings and the associated semantic data play key roles. The latter, in particular, includes qualitative environment information that can contribute to improving the robot's autonomy for navigation, task planning and manipulation, and simplifying human-robot interaction (HRI). This problem is tackled in the *Semantic Mapping* field, which aims to organize objects into classes and compute their pose and shape in a specific fixed reference frame. In this way, the environmental geometric information is supported by high-level features which increase the robot's awareness of the environment. In our specific case, we deal with the object detection and localization problem, which nowadays is widely investigated. For instance, in the last Darpa Subterranean Challenge[1], the main objectives were multi-robot exploration and object mapping in unknown environments, and the overall score was calculated based on the number of correctly detected and localized objects on the map.

[†]Intelligent and Autonomous Systems, Leonardo Labs, Genoa, Italy
[‡]HHCM & HRII, Istituto Italiano di Tecnologia, Genoa, Italy
[§]Industrial Innovation, DISI, Università di Trento, Trento, Italy
Authors' e-mail: {*name.surname*}*.ext@leonardo.com*

[1]Darpa Subterranean Challenge: https://www.subtchallenge.com/

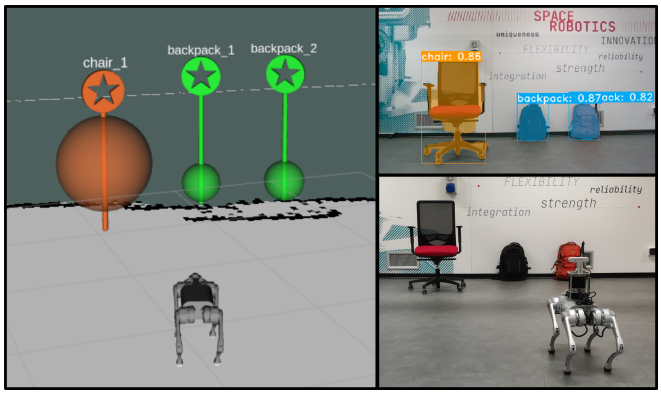

Fig. 1. An example of the framework during an experiment. On the left, is the visual application where objects are shown with a landmark and a spherical region of interest for the location in the Rviz visualization tool. On the top right, is the instance segmentation inference of the image taken from the robot camera while on the bottom right is the external representation of the experimental scene.

Different works were proposed to cope with the semantic mapping problem. Most recent results in robotics are facing the problem of using only RGB data and some interactive structures to be compliant with dynamic environments [1] while others rely on RGB-D data exploiting older algorithmic strategies (*e.g.* PnP algorithm) [2]. In autonomous driving, the RGB camera and lidar sensor fusion for semantic understanding is a currently tackled problem [3]. For a broader evaluation of the literature review see Sect. II.

Independently of the approaches used in robotics literature, the first thing which stands out is that most of them rely only on camera sensors. Cameras can give lots of dense information to the user especially if paired with depth data. However, their accurate depth range is within a few meters, leading to heavy depth measurement errors as the distances increase, especially if the robot is moving. This is particularly true for outdoor and vast indoor environments (*e.g.*, warehouses), where depth cameras are limiting and object semantic mapping remains a major challenge for far distances. In these cases, lidar sensors are an essential camera partner, allowing to have precise depth measurements for a wider distance range. Rather, in autonomous driving, the lidar and the RGB camera are nowadays commonly used but depth cameras are not considered due to their low resolution in the wide outdoor areas commonly faced in driving scenarios.

Another aspect not considered in most of the robotics examined works is that they do not account for limited re-

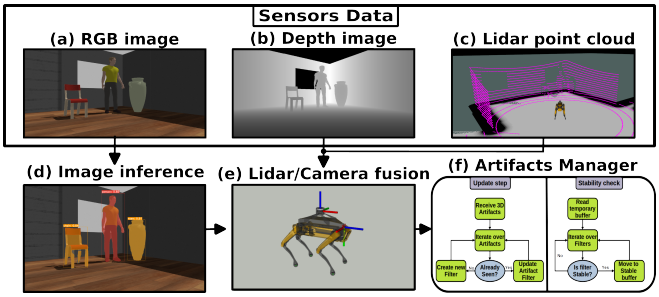

Fig. 2. This figure represents the whole Artifacts Mapping pipeline. The top block groups the sensors' data readings: (a) camera RGB image, (b) camera depth image and (c) lidar point cloud. At the bottom, there are (d) the RGB image inference performed with a Deep Neural Network for instance segmentation, (e) the multi-modal sensor fusion for detection and localization which uses as input the camera depth, the lidar point cloud and the Neural Network inference, and (f) a representation of the artifacts manager state-machines used to handle the sensor fusion detections and stabilize them.

sources applications which should run on embedded devices (*e.g.* Nvidia Jetson Nano[2]). Furthermore, Semantic Mapping is often used in the context of grasping or augmented reality scenarios while this work proposes an application for detecting and localizing objects (a.k.a artifacts) for high-level navigation tasks.

In our work, we aim to merge robotics and autonomous driving applications' strengths and present a modular architecture for semantic mapping[3]. We provide a multi-modal (camera-lidar) online semantic mapping framework which can fuse sensor information in real-time depending on the object distance and sensor's accuracies. We use image semantic information to enrich objects' filtered and stabilised positions to have precise object localization. The artifacts' shapes are simplified as spheres but they will be improved in future development. Our work relies on external geometric-based navigation frameworks such as SLAM algorithms or other localization algorithms (*e.g.*, AMCL [4]).

The proposed application demonstrates good accuracy for both near and far objects thanks to the camera-lidar depth fusion which, as far as the authors know, was not examined in other robotic or autonomous driving semantic mapping works. The application operates online also on low resources embedded systems (see Sect. IV-B) which strengthens the contributions of this paper. Moreover, we developed a Rviz[4] application which improves the user experience (UX) for visualization and interaction with the objects and the robot (see Fig. 1). The authors will provide on-demand a Docker application[5] as an added contribution, for running the artifacts mapping applications in simulation or on a robot (see Sect III-A-III-B and Sect. III-C).

The paper is divided as follows. In Sect. II a literature review of some recent works in semantic mapping is presented. The framework developed for this work is explained in Sect. III. We can distinguish the framework pipeline as two perception modules and a manager one. The first perception module performs 2D object detection while the second aims to estimate 3D artifacts position by fusing camera and lidar depth information (see Sect. III-A). The last module is needed to stabilize the perception estimations and to filter out noisy outliers (see Sect. III-B). An application of the presented framework (see Sect. III-C) is proposed based on two steps: *(i)* the robot can autonomously classify and localize objects on a map and save them in a specified format, *(ii)* the robot can load the artifacts as way-points on the map and the user can interactively select them to command the robot moving in that place to successively accomplish various kind of tasks such as manipulation, grasping, inspections or others. In Sect. IV the experiments to validate the framework are evaluated and discussed, and in Sect. V the conclusion and some future improvements are provided.

II. Related works

In literature, the semantic mapping problem was addressed using several approaches both in robotics and autonomous driving fields. Different surveys were presented that analysed this topic from various points of view. In [5] the authors explored the semantic mapping application in a human-robot collaboration scenario in an indoor environment while in [6] the semantic SLAM problem is presented in a general fashion analysing the works also in terms of perception, robustness, and accuracy. In [7], the less recent semantic mapping works are reviewed (*i.e.*, before 2014). This survey is a good reference to analyse the first development for the semantic mapping problem which yielded the more recent applications.

Among the modern semantic mapping approaches presented in robotics literature in the last decade, some first successful examples are [8] and [9]. In [8] the authors presented a monocular SLAM system that uses a SURF [10] feature extractor to check correspondencies and reconstruct the object's geometry. Instead, the authors in [9] showed an object-oriented 3D SLAM based on an ICP [11] object pose refinement and demonstrated that the introduction of semantic objects in the SLAM loop improves performances. the authors in [12] developed a monocular SLAM-aware object recognition system based on multi-view object proposals and efficient feature encoding methods giving as output a semi-dense semantic map. In [13] the authors proposed a framework which directly manages 3D objects. They use a Kinect[6] camera to reconstruct the 3D environment from different points of view and classify them while estimating their pose. In [14] the Data Associated Recurrent Neural Networks (DA-RNN) is introduced, which is an RNN for semantic labelling of RGB-D videos. The network output

[2]Nvidia Jetson Nano: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[3]Artifacts Mapping Youtube videos: https://www.youtube.com/playlist?list=PLdibjJfM06zugiWd-yUcdGH-SRWKTA3nQ

[4]rviz: http://wiki.ros.org/rviz

[5]The authors will grant access to a Docker image with the compiled application upon acceptance of the paper, based on the Freeware license.

[6]Microsoft Kinect camera https://en.wikipedia.org/wiki/Kinect

is fused with the KinectFusion algorithm [15] to merge semantic and geometric data. In [16] a Convolutional Neural Network (CNN) is used along with the ElasticFusion SLAM algorithm [17] to provide long-term dense correspondences between RGB-D video frames even in loopy trajectories. The authors in [18] leveraged ORB-SLAM2 [19] to reconstruct the geometric environment while using Single-Shot multi-box Detector (SSD) [20] along with an unsupervised 3D segmentation algorithm to place objects in the environment.

Moving towards more recent works, in [21] is presented the Contextual Temporal Mapping (CT-Map). They modelled the semantic inference as a Conditional Random Field (CRF) to account for contextual relations between objects and the temporal consistency of their pose. MaskFusion [22] is a real-time object-aware semantic and dynamic RGB-D SLAM algorithm. The greatest difference with respect to its predecessors is that it can cope with dynamic objects by continuously labelling them. Fusion++ [23] performs an object-level SLAM based on a 3D graph map of arbitrary reconstructed objects. They used RGB-D cameras, Mask-RCNN [24] instance segmentation and the Truncated Signed Distance Function (TSDF) to perform the semantic reconstruction. In [25] is presented an approach that incrementally builds a volumetric object-centric map with an RGB-D camera. They used an unsupervised geometric approach with instance-aware semantic predictions to detect previously unseen objects. They then associated the 3D shape locations with their classes if available and integrate them into the map. This approach has limited time performances to be used on a mobile robot because it runs at 1 HZ so it could be impractical in real-time. Conversely, in [26] the authors obtained a real-time dense reconstruction and semantic segmentation of 3D indoor scenes. They used an efficient super-voxel clustering method and conditional random fields (CRF) with higher order constraints from structural and object cues, enabling progressive dense semantic segmentation without any precomputation. The CRF infer optimal segmentation labels from the prediction of a deep neural network and runs in parallel with a real-time 3D reconstructor which utilizes RGB-D images as input. In [27] an open-source C++ library for metric-semantic visual-inertial SLAM in real-time is presented. They provide a modular code composed of a visual-inertial odometry (VIO) module, a pose graph optimizer, a 3D mesh-building module, and a dense 3D metric-semantic reconstruction module. The authors in [28], used a UAV equipped with a lidar, an RGB camera and a thermal camera to augment 3D point clouds and image segmentation masks while also generating an allocentric map.

One of the last available works which focus on this topic is [1] which presented a semantic mapping framework which uses only RGB data. They did not accomplish only object mapping but they provided a framework that can also distinguish different rooms and buildings. They exploited the 3D dynamic scene graphs [29] to abstract the different layers of inference (*i.e.* object, room and building), to solve problems such as loop closure detection and to cope with the mapping problem. Instead, the authors of [2] used RGB-D cameras to reconstruct an allocentric semantic map. They used a keypoint-based approach for pose estimation using a CNN keypoint extractor trained on synthetic data. Object poses were recovered from keypoint detections in each camera viewpoint with a variant of the PnP algorithm. The outputs obtained from the multi-camera system were then fused using weighted interpolation.

In autonomous driving, the multi-sensor fusion problem for 3D object detection is faced in [3] which uses lidar and RGB camera sensors to estimate the objects positions in the environment through ground estimation and depth completion. They use an end-to-end approach to train their multi-task network. The authors in [30] build a semantic map with a laser-based semantic segmentation of the point cloud not requiring any camera data. In [31], the authors provided a lidar-based SLAM for the geometric mapping and then use a CRF to fuse and optimize the camera semantic labels to obtain the semantic map. Instead, in [32], the camera and lidar data are used to build a probabilistic semantic octree map considering all the uncertainties of the sensors involved in the process. The authors in [33] presented one of the latest works in autonomous driving semantic mapping. They use an RGB camera and a lidar to perform semantic segmentation, direct sparse visual odometry and global optimization to include GNSS data in the mapping process.

Our review of the state-of-the-art indicated that most of the works on robotics platforms rely only on camera measurements and the experiments are limited to small indoor environments. Instead, in the autonomous driving scenario the camera-lidar fusion is already used for semantic tasks but they rarely use depth cameras, their lidars are generally more powerful (*i.e.*, they have 128-row lidars compared to the 16 ones commonly used in robotics) and they test the application in driving outdoor scenarios which offer different challenges with respect to robotic indoor once. Hence, with our work, we aim to stress the fact that RGB-D cameras and lidars are complementary sensors also in robotic semantic applications. For the semantic mapping application, we stated that with both sensors we can correctly localize objects at different distance ranges, improving detection accuracy.

III. Artifact Mapping Framework

In this section, the whole framework is presented as a conjunction of two blocks: Sect. III-A for object perception and Sect. III-B for object managing. In Sect. III-C the provided UI application is illustrated.

A. Artifacts detection and position estimation

The perception part can be conceptually divided into two components: *(i)* 2D object segmentation, *(ii)* 3D object position estimation using camera-lidar filtering.

1) 2D object segmentation: In this phase, a deep neural network [34] is used to infer from RGB images (see Fig. 2a) some predefined objects' classes and their masks. During the navigation, the robot takes pictures of the environment using the camera mounted on it. The pictures are passed

into an instance segmentation deep neural network which outputs the classification labels and masks (*i.e.*, a binary image having 1 where the object is found and 0 elsewhere) for each object recognized on the image (see Fig. 2d). The outputs are grouped and passed to the next module which will convert 2D data into 3D ones. An optional feature provided in this module is the possibility to filter out classes in real-time upon request. In this way, the robot can map different objects online depending on the requirements proposed. Other implementation aspects will be further explained in Sect. IV.

2) 3D object position estimation using camera-lidar filtering: This module fuses RGBD camera and lidar measurements to have a precise estimate of the objects' positions in the environment. The input is composed of the classification labels and masks found in the previous module, and depth information extracted from the camera (see Fig. 2b) and the lidar (see Fig. 2c). Sensors depth measurements are first analyzed separately in the following.

The depth image obtained from the camera (see Fig. 2b) is filtered using the recognized objects masks through element-wise matrix multiplication. The output, containing only the depth data of the object plus some sensor noise and environment outliers, is used to build a 3D point cloud projecting the 2D image points in the 3D space using the formula in the equation:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{p_x}{f_x} \\ 0 & \frac{1}{f_y} & \frac{p_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} z_C \ , \tag{1}$$

where x_C, y_C, z_C are the 3D point coordinates with respect to the camera, u, v are the pixels on the image plane and f_x, f_y, p_x and p_y are the camera intrinsic parameters (focal distances and sensor's centre). Note that z_C is the depth measured by the camera depth sensor.

The obtained point cloud is filtered using a voxel grid downsampling filter[7] to reduce the number of points and, consequently, a radius outlier filter[8] is applied to remove the outliers induced by sensors noises and inference imperfections. The final point cloud is then used to compute the camera artifact centroid X_C as the mean of its points.

The 3D lidar centroid estimation is computed as follows. Projecting the 3D lidar points (see Fig. 2c) in the 2D detected masks images using Eq. 2, we are able to extract the object points of interest from the point cloud (*i.e.,* the points which have the 2D projection inside the mask).

$$z_L \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_L^C & T_L^C \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} \ , \tag{2}$$

where $R_L^C \in \mathbb{R}_{3x3}$ and $T_L^C \in \mathbb{R}_{3x1}$ are the rotation matrix and the translation vector between the lidar and the camera,

[7]voxel grid downsampling filter: https://pointclouds.org/documentation/tutorials/voxel_grid.html

[8]radius outlier removal: https://pointclouds.org/documentation/tutorials/remove_outliers.html

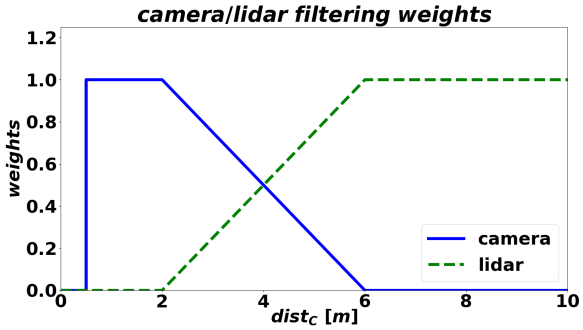

Fig. 3. An example of the contribution weights of camera and lidar for sensor fusion. The camera weight is in blue while the lidar one is in dashed green. In the specific example, we considered the specifications of a generic RGB-D camera you can find on the market: $min_C = 0.5$, $acc_C = 2.0$ and $max_C = 6.0$.

x_L, y_L, z_L are the 3D centroid position with respect to the lidar and the other parameters are the same of Eq. 1.

The extracted point cloud, representing the noisy artifact, will be then filtered using a radius outlier filter similar to the one used for the camera. Both radius filter parameters are directly dependent on the number of point cloud points because different distances and sizes of objects affect the point-cloud density and consequently the filtering. Finally, the mean of the point cloud is computed to obtain the lidar artifact centroid X_L.

Once both centroid measurements are available, they are fused in the artifact centroid X following the rules in the equation:

$$X = \begin{cases} 0 & \text{If } dist_C < min_C \\ X_C & \text{If } min_C \leq dist_C \leq acc_C \\ \xi X_C + (1-\xi) X_L & \text{If } acc_C \leq dist_C \leq max_C \\ X_L & \text{If } dist_C > max_C \ , \end{cases} \tag{3}$$

where $dist_C$ is the euclidean distance between the 3D point estimates and the camera, min_C and max_C are the minimum and maximum distances the depth camera can perceive, acc_C is the distance within which the camera can have accurate enough measurements to be used alone for the object localization (the camera information are generally provided by the sensors vendors), $X_L \in \mathbb{R}^3$ and $X_C \in \mathbb{R}^3$ are the lidar and camera 3D centroid estimates and $\xi \in [0,1] \in \mathbb{R}$ is the fusion weight represented by the blue slope of the segments between acc_C and max_C in Fig. 3 and it is computed as follows:

$$\xi = -\frac{1}{max_C - acc_C}(dist_C - acc_C) + 1 \tag{4}$$

Using the filtered camera and lidar point clouds, a rough 3D radius estimation ρ of the objects is performed. The camera radius ρ_C and the lidar radius ρ_L are computed as the mean of the two bigger dimensions along the X, Y and Z point cloud axis. the final radius ρ is computed following the same centroid fusion rules of Eq. 3 substituting X with ρ, X_C with ρ_C and X_L with ρ_L.

Also, the view angle ϕ of the artifact with respect to the robot is computed. Such an angle is rotated with respect to the map reference frame for implementation reasons with equation:

$$\phi = atan2(r_{21}, r_{11}) + atan2(y_r, x_r) \ , \tag{5}$$

where the r_{ij} is the entry at row i and column j of the rotation matrix $R_r^m \in \mathbb{R}_{3x3}$ between the map m and the robot r and x_r, y_r are the x, y positions of the artifact centroid with respect to the robot base. The two addends of Eq. 5 represent respectively the heading angle between the robot and the map and the angle between the robot and the 3D centroid.

B. Artifacts manager for data association

The manager (see Fig. 2f) is needed to filter out outliers and to stabilize artifact position estimations provided by the sensor fusion module. This process is generally known as data association[5][14]. The manager is composed of two modules: *(i)* object position filtering and *(ii)* object position stabilization which runs asynchronously in parallel.

1) Position filtering: Using a temporary data structure, the *temporary buffer*, we store and filter the perceived artifacts. Once the manager receives the 3D artifacts position estimations from the perception module (see Sect. III-A), it checks if the artifacts were already seen before (*i.e.,* the distance between one of the already seen artifacts and the current one is less than its 3D radius). If this is the case then the artifact in the temporary buffer is updated. Otherwise, for each not previously seen artifact received, the manager creates a new artifact instance in the temporary buffer. These instances have their own moving average filter which estimates the average of the artifact centroid position and its radius with Eq. 6 and computes a variance based on the distances between the position and the moving average in the filter horizon with Eq. 7.

$$\mu = \frac{1}{N} \sum_{\chi \in \Omega_N} \chi \tag{6}$$

$$\sigma = \frac{1}{N} \sum_{\chi \in \Omega_N} ||\chi - \mu||^2 \ , \tag{7}$$

where $N \in \mathbb{N}$ is the number of measurement in the moving average set Ω_N of 3D points, $\chi \in \mathbb{R}^3$ represent the current 3D position measurement, $\mu \in \mathbb{R}^3$ is the 3D mean position and $\sigma \in \mathbb{R}$ represent the variance of the filter.

2) Position stabilization: This module checks the stability of the artifacts in the temporary buffer and stores stable artifacts in another similar structure, the *stable buffer*. If an artifact in the temporary buffer is stable, the stabilizer moves the artifact from the temporary buffer to the stable one. An artifact is considered stable when its moving average filter variance σ is less than half its 3D artifact radius ρ and at least half the average filter set Ω_N is filled. This means that we have enough stable object position estimations and the object position average can be used for fixing the object position on the map.

At the end of the Artifacts Mapping application, an additional data association step is performed. The artifacts belonging to the same class which overlay each other on the XY plane are merged into a single artifact. This step reduces the duplicated object which sometimes appears on the map due to different point-of-view measurements and occlusions. After that, the stable artifacts buffer is saved in a yaml file which could be loaded into the user interface application presented in the next section.

C. User Interface for goal sending

A User Interface (UI) application based on a Rviz plugin (see Fig. 1) was developed to provide an intuitive visualization of the artifacts on the map, to send commands to the robot for moving near an artifact of interest and to delete artifacts which the user do not need or are wrong. Such artifacts can be loaded from the yaml file obtained with the artifacts mapping application. Through the UI application, the user can send *nav_msgs/goal* ROS messages which can be used by the robot to move towards the object (*e.g.*, using the ROS navigation stack as we do, see Sect. IV). The user can interact with the artifacts by simply right-clicking on them on Rviz and selecting the action *Go To* or *Delete*. Being the artifacts centroid position inside the artifacts shapes, the goal is moved in front of the artifact so that the robot stops before colliding with the object. The other available option is artifact deletion. If the user notices that an artifact is wrongly identified (classification or position) then the user can delete it and, once the UI application is closed, the loaded yaml file is updated with the remaining artifacts.

IV. Experiments

The experiments are performed both in simulation and using a real robot in a laboratory environment. The experimental setup is the same: some chosen objects are randomly positioned in the experiment area and the robot, following a predefined path, maps the predefined objects it encounters. This strategy is chosen because the objective is the validation of the artifacts mapping accuracy during an application, for example during a patrol. In other application scenarios, *e.g.* search and rescue, our framework could run in parallel with an exploration algorithm and the robot could trigger the exploration module every time an object of interest is encountered to obtain a precise localization.

In the experiments, we compare the data fusion with the mono-sensors application (i.e. using only an RGB-D camera or only the lidar) to demonstrate that the data fusion highly improves the detection accuracy and decreases the errors. For each environment setup, the experiments are repeated three times, one for each *sensors configuration*: only camera, only lidar, and both.

This work focuses only on semantic mapping and does not account for the robot localization which is assumed to be given. Additional errors in mapping resulting from localization are not considered in the final evaluation even if

TABLE I

DETECTION RESULTS OF THE SIMULATION AND REAL EXPERIMENTS.

	Simulation			Real		
	Camera	*Lidar*	*Fusion*	*Camera*	*Lidar*	*Fusion*
Correct detection	386	391	**416**	86	81	**99**
Wrong localization	12	13	**7**	10	14	**2**
Duplication	19	24	**15**	10	14	**6**
Wrong classification	**0**	**0**	**0**	7	11	**6**
Total detections	417	428	433	113	120	113
Total objects	422			101		

they negatively affect our application. Moreover, is important to notice that quadrupedal robots' movements are jerky and the sensors can suffer from that.

We set the parameters min_C, acc_C and max_C of Eq. 3 as 0.3, 4, 6 respectively based on the camera hardware information provided by the camera vendors (Intel Realsense).

The final validation performance is based on the number of objects which the robot can correctly find over the number of total objects. Also, the number of correctly-detected objects over the total number of detections is evaluated. The object is considered *found* if the difference between the estimated position and the real one is less than the real object radius and the associated class label is correct. The errors are categorized as duplicated objects, wrong localization and wrong classification. The duplications occur when there are more artifacts on a single object. they could be caused by the wrong artifacts radius computation due to occlusions or distinct point of view detection (*i.e.,* viewed from different perspectives: front and behind). The localization is considered wrong if the artifact's estimated position is outside the real object shape while the classification is erroneous if the artifact's class label is not correct.

For the simulation, the Whole-body Locomotion Framework (WoLF)[35] is used on a notebook with an *Intel® Core™ i9-11950H* processor and an *NVIDIA Geforce RTX 3080 Laptop* GPU. In the real scenario, a Unitree Go1[9] quadrupedal robot equipped with a RoboSense RS-Helios16 lidar[10], an Intel RealSense D455[11] and three Nvidia Jetson[12] (two Jetson Nano 4GB and one Nvidia Xavier NX) are used for the evaluation. The experiments are performed with the instance segmentation algorithms Yolact++ [34] and YolactEdge [36] trained on COCO [37] data set.

A. Simulation Experiments

Gazebo[13] simulator is used to simulate the robot in two different environments: the office[14] and Maze worlds where a predefined number of objects are positioned randomly at each iteration. The chosen objects for the simulation evaluation are *vase*, *couch*, *plant* and *person*. Specifically, in the office world, there are 5 vases, 12 couches, 6 plants and 11 persons while in the Maze world, there are 15 vases, 13 couches, 12 plants and 12 persons. The robot path is chosen randomly in advance using some waypoints on the map. In total, for each *sensors configuration*, 10 experiments were conducted, 5 for each environment, using different setups, for a total of 30 experiments.

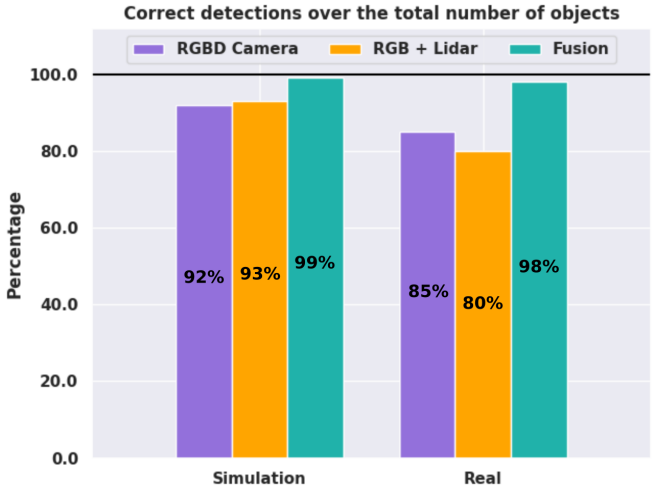

Fig. 4. Percentage of the correctly mapped and labelled objects concerning the total number of objects on the scene. On the left are the simulation results and, on the right, are the real experiments. Each block has three histograms representing the three *sensors configurations* used during the experiments: only RGBD camera, only RGB + lidar, and both.

The results of the simulation experiments are shown in the left part of Fig. 4 in terms of the number of correct detected objects. Specifically, considering the three ordered *sensors configurations* (*i.e.* only camera, only lidar, and both), we obtain the 92%, 93% and 99% of correctly localized and classified objects. Moreover, analysing the total number of detections produced, we obtain the distribution of the detections represented in the left column of Table I and the top part of Fig. 5 for the simulation experiment. Among all the detection produced, considering again in order the three *sensors configurations*, the 92%, 91% and 95% were correct while the remaining 8%, 9% and 5% of them were wrong.

The farthest object correctly detected in simulation during the camera-lidar sensor fusion experiments was at $15.47m$ from the robot, while the nearest was at $1.23m$.

[9]Unitree Go1: https://www.unitree.com/en/go1/

[10]RoboSense RS-Helios16: https://www.robosense.ai/en/rslidar/RS-Helios

[11]Intel RealSense D455: https://www.intelrealsense.com/depth-camera-d455/

[12]Nvidia Jetson: https://www.nvidia.com/it-it/autonomous-machines/embedded-systems/

[13]Gazebo simulator: https://gazebosim.org/home

[14]Clearpath robotics worlds: https://github.com/clearpathrobotics/cpr_gazebo/tree/noetic-devel/cpr_office_gazebo

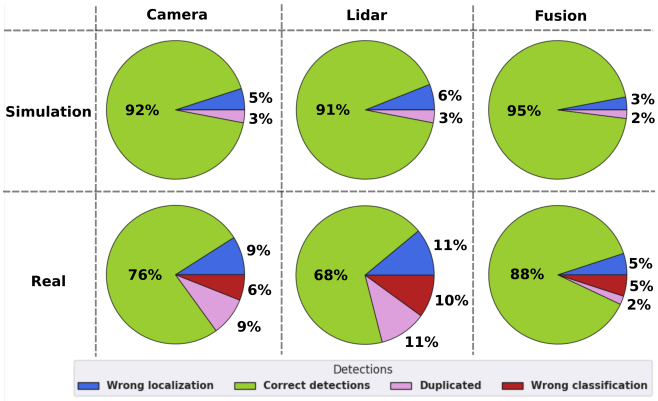

Fig. 5. Distribution of correctly and wrongly detected artifacts among the total generated detections. The pie charts represent the distribution of the correctly detected artifacts in green, the doubled objects in blue, the wrongly localized ones in pink and the wrongly classified ones in red. On the top row are the simulations results while on the bottom are the real ones. For each row, experiments are divided into three columns depending on the *sensors configuration* used during experiments: only camera, only lidar, and both.

B. Laboratory Experiments

The real experiments were carried out in a laboratory setting considering two scenarios, a one-room laboratory environment and a complete floor environment where the robot can move through corridors. In these environments were positioned *umbrellas*, *chairs*, *cabinets*, *backpacks* and *TVs* in variable amounts. For each *sensors configuration*, A total of 6 experiments were conducted, 3 for each environment, for a total of 18 experiments. For each trial, the objects were randomly moved and the illumination changed, *i.e.*, switching off lights or closing shutters.

The results of the laboratory experiments are shown in the right part of Fig. 4 in terms of the number of correct detected objects. Specifically, considering the three *sensors configurations* in order (*i.e.* only RGBD camera, only RGB + lidar, and both), we obtain respectively the 85%, 80% and 98% of correctly localized and classified objects. Moreover, analysing the total number of detections produced, we obtain the distribution of the detections represented in the right column of Table I and the bottom part of Fig. 5 for the real experiment. Among all the detection produced, the 76%, 68% and 88% were correct while the remaining 24%, 32% and 12% of them were wrong.

The farthest object correctly detected during the camera-lidar sensor fusion experiments was at a distance of $10.37m$ from the robot, while the nearest was at $0.98m$.

C. Discussion

The first thing to point out is that the farthest distances of the detected object were greater than $10m$ both in simulation and in real experiments. We take into account this distance to show a qualitative comparison between the lidar and RGB-D measurement in Fig. 6. The figure qualitatively upholds the thesis that a lidar sensor along with the camera is necessary to improve semantic mapping and, in general, other detection algorithms in wide areas. Moreover, from the results obtained

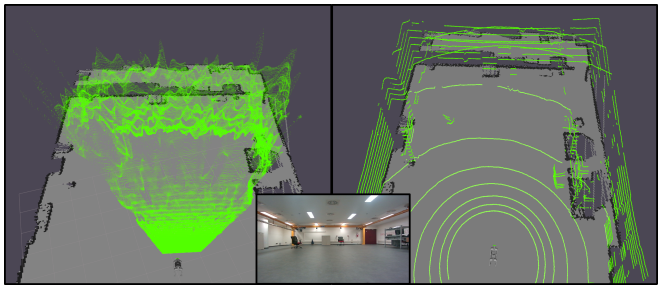

Fig. 6. Qualitative comparison between RGB-D camera (left image) and lidar (right image) point cloud detections at an approximate distance of $10m$ from the wall. At the bottom centre, there is the representation of the scene taken with the robot camera at that time instant. At large distances, the camera data are noisier and less accurate with respect to the lidar one but at small distances cameras provide a denser accurate point cloud while lidar data are sparser. From this comparison can be deduced that a visual-lidar sensor fusion can enhance semantic mapping.

from the experiments, it is clear that in our framework the use of both sensors improves the robustness of the application and decreases the detection errors. These improvements are less evident in a simulation environment where we used almost ideal sensors, *i.e.* the noise representation is not realistic as in Fig. 6. Still, it impacts real scenarios where there is more sensor noise.

The lidar can map far obstacles precisely while the camera introduces lots of errors at high distances. If we adopt only the camera, one solution to avoid erroneous measurements could be to not consider the depth measurement out of the accurate range guaranteed by the device specifications. By the way, by doing this the robot could miss some artifacts if it does not get close enough to them.

The camera, by providing more information at near distances with respect to the lidar, yields more precise centroid computations because it has fewer outliers than the lidar. Lidar outliers can be caused by wrong camera-lidar pose calibration and time synchronization which are essential for these applications especially when the robot moves fast. Instead, with RGBD cameras, the depth and the RGB images are synchronized in time and can be spatially superimposed almost exactly.

It is important to notice that wrong classification errors result from erroneous classifications in the pre-trained instance segmentation neural network which can be caused by illumination, reflections or other environmental conditions. They are here considered because the image inference is a module of the proposed pipeline but such errors can be decreased using more powerful neural networks.

V. Conclusion

We presented a framework which uses multi-modal sensors fusion to tackle the semantic mapping problem which is a rare setup in robotics applications. We fuse the lidar and RGB-D camera sensor readings to achieve better accuracy both for near and far objects as opposed to camera-only systems which lose accuracy for distant objects or lidar-only which lack high-level texture understanding of the environment.

We proposed a UI application to interact with the artifacts map obtained during the mapping application. This application is useful to perform autonomous high-level decision-making tasks because it exposes the object's class and location to the robot and the user.

The experiments showed that our application can correctly detect, localize and map the 98% of the objects present in the scene at different distances providing a small number of detection errors and good localization accuracy. The comparisons with the single-sensor scenario (only camera or only lidar) proved that sensor fusion is essential for wide areas and high-accuracy applications.

There are different future improvements we planned for this framework: (i) evolve the algorithm to an independent graph-based SLAM system, (ii) use 3D semantic point clouds with oriented bounding boxes and dimension information for better visualization and object understanding, (iii) deal with dynamics obstacle.

REFERENCES

[1] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," *Robotics: Science and Systems XVIII*, 2022.

[2] J. Hau, S. Bultmann, and S. Behnke, "Object-level 3d semantic mapping using a network of smart edge sensors," *arXiv preprint arXiv:2211.11354*, 2022.

[3] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[4] W. Xiaoyu, L. Caihong, S. Li, Z. Ning, and F. Hao, "On adaptive monte carlo localization algorithm for the mobile robot based on ros," in *Chinese Control Conf (CCC)*, 2018.

[5] A. Achour, H. Al-Assaad, Y. Dupuis, and M. El Zaher, "Collaborative mobile robotics for semantic mapping: A survey," *Applied Sciences*, 2022.

[6] L. Xia, J. Cui, R. Shen, X. Xu, Y. Gao, and X. Li, "A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots," *International Journal of Advanced Robotic Systems*, 2020.

[7] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, 2015.

[8] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic slam using a monocular camera," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, 2011.

[9] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, 2008.

[11] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, 1992.

[12] S. Pillai and J. Leonard, "Monocular slam supported object recognition," *arXiv preprint arXiv:1506.01732*, 2015.

[13] K. Tateno, F. Tombari, and N. Navab, "When 2.5 d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam," in *2016 IEEE international conference on robotics and automation (ICRA)*, 2016.

[14] Y. Xiang and D. Fox, "Da-rnn: Semantic mapping with data associated recurrent neural networks," *arXiv preprint arXiv:1703.03098*, 2017.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*, 2011.

[16] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*, 2017.

[17] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," in *Robotics: Science and Systems*, 2015.

[18] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[19] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, 2017.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2015.

[21] Z. Zeng, Y. Zhou, O. C. Jenkins, and K. Desingh, "Semantic mapping with simultaneous object detection and localization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[22] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018.

[23] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 international conference on 3D vision (3DV)*, 2018.

[24] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[25] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3d object discovery," *IEEE Robotics and Automation Letters*, 2019.

[26] Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Real-time progressive 3d semantic segmentation for indoor scenes," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[27] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[28] S. Bultmann, J. Quenzel, and S. Behnke, "Real-time multi-modal semantic fusion on unmanned aerial vehicles," in *2021 European Conference on Mobile Robots (ECMR)*, 2021.

[29] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.

[30] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[31] J. Li, X. Zhang, J. Li, Y. Liu, and J. Wang, "Building and optimization of 3d semantic map based on lidar and camera fusion," *Neurocomputing*, 2020.

[32] J. S. Berrio, M. Shan, S. Worrall, and E. Nebot, "Camera-lidar integration: Probabilistic sensor fusion for semantic mapping," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[33] Q. Cheng, N. Zeller, and D. Cremers, "Vision-based large-scale 3d semantic mapping for autonomous driving applications," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[34] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact++: Better real-time instance segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[35] G. Raiola, M. Focchi, and E. M. Hoffman, "Wolf: the whole-body locomotion framework for quadruped robots," *arXiv preprint arXiv:2205.06526*, 2022.

[36] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee, "Yolactedge: Real-time instance segmentation on the edge," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014.