Semantic Map Augmentation for Robot Navigation: A Learning Approach based on Visual and Depth Data

Anonymous LARS 2018 submission

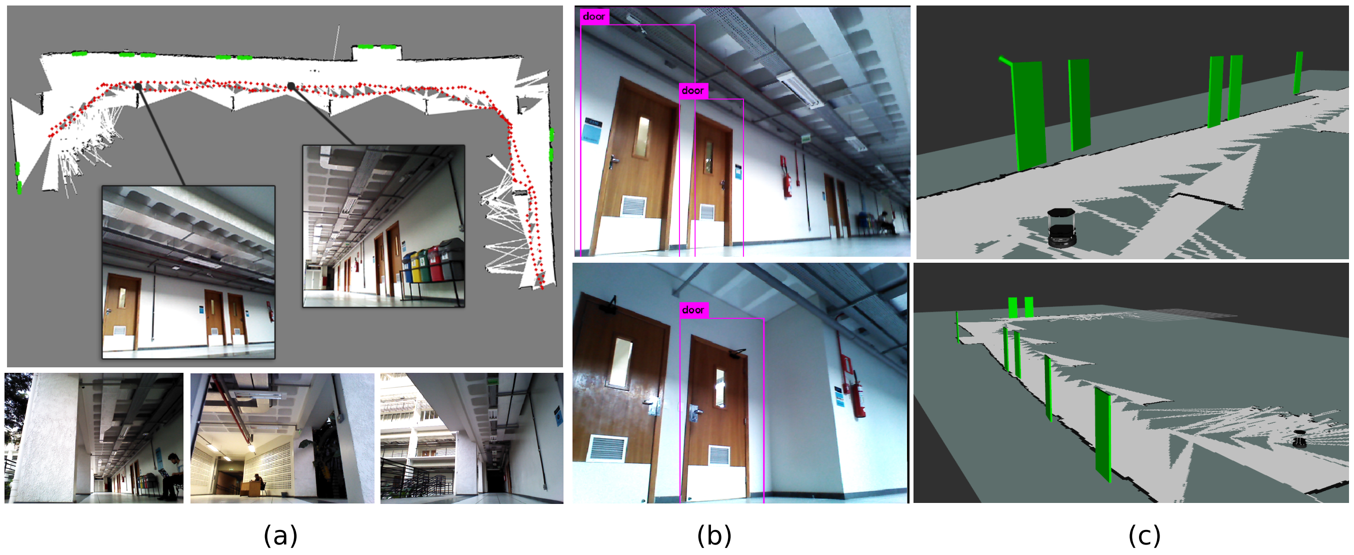

Fig. 1. Augmented semantic mapping overview. (a) Bird's-eye view of the 2D map and door locations (in green) and some image frames of the dataset; (b) Object detection examples; and (c) Visualizations of the augmented semantic map output.

***Abstract*— In this paper, we propose a framework to build an improved metric representation of the environment with semantic information. We explore some of the recent advances of deep neural networks to object detection/semantic classification in a visual-based perception scheme. The output of this system is a map of the environment extended with semantic object classes and their positioning. This framework combines sensors available in commonly used mobile robotic platforms such as an RGB-D camera, 2D LIDAR and odometers. In short, a CNN-based object detector and a 3D model-based segmentation technique are used to localize and identify different classes of objects in the scene. Then the tracking of the semantic classes is performed with a Kalman filter approach. We show results for "door" objects and validate this approach with a collected dataset in an extensive indoor area, comprising corridors and offices. A new dataset and the source code are made available to the community as ROS packages[1].**

I. INTRODUCTION

Scene understanding is a crucial factor in the development of robots that can effectively act in an uncontrolled, dynamic, unstructured, and unknown environment, such as those found in real-world scenarios. Several tasks in the field of mobile robotics such as long-term mapping, obstacle avoidance, and autonomous navigation have received a wide attention in recent years, but they are still often challenged by adversities found in those environments. In this context, a higher level of understanding of the scene (such as identifying certain objects, people, as well as localizing them) is usually required to perform effective navigation and perception tasks. Typical examples are self-driving cars, which need to recognize other vehicles, pedestrians and the free road space for safe navigation [1], [2], or personal assistant robots that need to recognize the environment and humans for safe interaction [3], [4]. Recent advances on the computing capability of embedded devices and artificial intelligence techniques have allowed this higher level of understanding to be comparable to human-like performance. It is then interesting to integrate these advances into robotics systems, allowing them to perform more complex tasks in less specialized environments.

In this paper, we propose an open framework for building hybrid maps, combining both environment structure (metric map) and environment semantics (objects classes) to support autonomous robot perception and navigation tasks. Applications of this representation are in place categorization, contextual navigation or in "situation awareness" by distinguishing dynamic objects (e.g., people) from static ones (e.g., doors, walls, windows), to name a few. We propose a framework that detects and models objects in the environment from RGB-D images using convolutional neural networks to capture higher-level information. Finally, the metric map is augmented with the semantic information extracted using the object categories. Moreover, we also provide a new dataset acquired in an extensive indoor environment comprising corridors and offices with RGB-D images, 2D LIDAR and

[1]The dataset and source code repositories links are not included now in the paper to respect the double-blind review process.

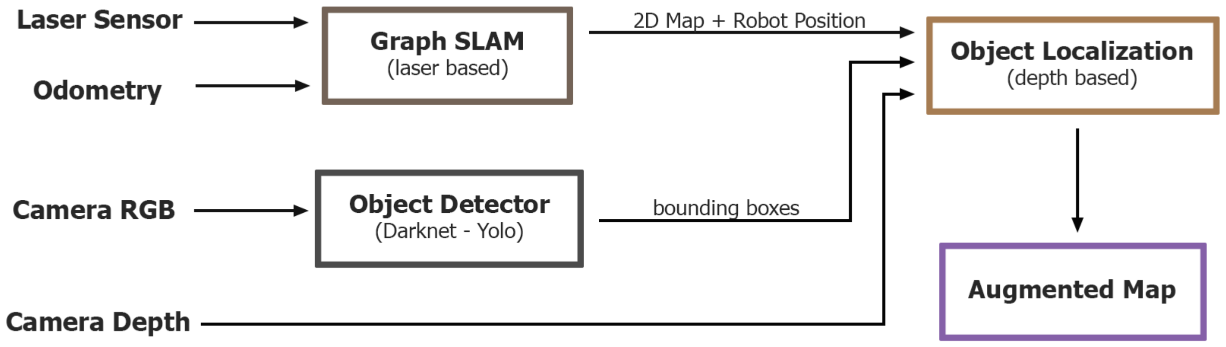

Fig. 2. Visualization of the formulation pipeline, showing the main modules and some of the information exchanged between them.

odometry data. Figure 1 shows some examples of data from our dataset and the result of our mapping framework.

The reminder of the paper is organized as follows. First, Section II presents some recent related works. Next, we describe our semantic map augmentation in Section III and the proposed dataset in Section IV. The experiments and implementation details are presented in section V. Finally, Section VI concludes the paper and discuss some perspectives.

II. RELATED WORK

Object detection and semantic segmentation are challenging computer vision problems. These problems are relevant to several tasks in diverse domains (e.g., virtual reality, robotics, intelligent vehicles) and have seen outstanding advances from recent deep learning techniques [5], [6], [7], [8]. Most techniques generate region proposals, i.e., bounding boxes, for possible objects in the image; predicting the most likely class for each region and suppressing overlapping regions while discarding regions with low probability scores.

Semantic mapping and localization have seen a trend in recent years. For instance, Nascimento et al. [9] applied a binary RGB-D descriptor to feed an Adaboost learning method to classify objects in a navigation task. McCormac et al. [10] propose a method for semantic 3D mapping. Their work combine the work of Whelan et al. [11], an RGB-D based SLAM framework for reconstructing the scene into a 3D dense point cloud, with a convolutional/deconvolutional neural network for semantic segmentation. Projecting the segmented labels onto the 3D reconstructed point cloud allows for semantically labeling each 3D point.

In a similar fashion, the work of Li and Belaroussi [12] also aims at providing a method for 3D semantic point cloud mapping, but without monocular images. Their methodology uses LSD-SLAM, which provides a semi-dense 3D reconstruction and localization system from monocular images [13]. Likewise, the combined point cloud and projected segmented image provides a semantic 3D map. Our work follows the trend of these approaches to build more elaborated representations of the environment, taking advantage of the recent advances in machine learning and RGB-D sensory technologies. A similar methodology outline to ours can be seen in the Intel Realsense Object Library for RGB-D cameras [14]. They propose an algorithm that performs object detection using the RGB camera information, object localization and finally object tracking through different time frames. However, it is designed for reactive navigation and does not support any kind of environment mapping such as the one proposed in this work.

III. METHODOLOGY

Our augmented scene representation approach can be divided into three main modules: a SLAM module, which creates a 2D map of the environment using the on-board robot sensors; the object detector module, which runs a convolutional neural network that detects pre-trained object classes in real-time; and an object positioning module that processes the information of the previous modules with the point cloud information of the RGB-D image to localize and track previously localized objects in the map. Figure 2 depicts an overview of our approach.

A. Localization and Mapping

In this section, we briefly describe the first stage of the augmented mapping framework. The localization and mapping module gives a 2D representation of the environment, along with the localization of the robot in this representation. We select the standard Gmapping SLAM algorithm [15]. This algorithm produces a grid-based map using depth information (we denote this map in the text as $\mathbb{M}$). Each cell of this grid represents a small region of the space, and will be classified as either free space, occupied or unknown. The algorithm uses Rao-Blackwellized Particle Filters to perform the mapping and localization. The output is the 2D grid-based representation of the environment $\mathbb{M}$ along with the location $\mathbf{x} \in \mathbb{R}^3$ of the robot in the map:

$$\mathbf{x}_r = (x,\ y,\ \theta)^T, \tag{1}$$

where (x, y) is the position and θ the orientation. It is worth noting that one can use any available SLAM algorithm, such as only an RGB-D camera [16] or 2D laser [17].

B. Object Detection and Localization

Concurrently to the localization and mapping module described in Section III-A, we perform the detection of the object classes of interest that are in the field-of-view of the RGB-D camera. For that, we profit of recent advances on

CNN-based techniques to detect objects and infer the semantic information, i.e., the locations and classes of objects. We selected the "You only look once" (Yolo) system [8] among the various available object detection techniques, because of its low computational effort and high precision recall. Unlike typical object detection algorithms (such as the recent Faster-RCNN [6]) that generally decouple the detection into a large set of classification tasks (using CNNs as ResNet [18] or AlexNet [5]), Yolo is based on a network model called Darknet-19 that outputs not only the probability distribution of the classification of each object class, but also the bounding boxes for the objects themselves. The processing is faster, because it only takes a single network evaluation for each image to generate all the boxes and probabilities. In short, Yolo divides the input image into a grid with 13×13 cells. Each cell contains the center of five different bounding boxes proposals, where each bounding box has four parameters (position (x, y), width and height). The final network layer will have an output of shape $13 \times 13 \times 5(N+5)$, i.e., the probability distribution for all cells considering N object classes. Finally, only those boxes whose probability overcome a certain threshold are accepted (usually around 0.30).

In the context of indoor scenes, the training images of Yolo contain classes such as "door", "chair", "person" and "window". The final output of the system is the object boxes (encoded by five parameters) at a frame-rate of 15Hz in a typical PC. Some detection examples can be seen in Figures 1 and 3.

Model Fitting and Localization

This section describes the localization of the detected objects and the respective object model parametrization. Once the objects are detected by Yolo, we localize and fit a primitive 3D model related to each object class. For instance, a simple convex hull such as a 3D cylinder primitive can be used for representing "person" or "chair" classes; and a plane is a reasonable primitive for representing "door". The latter is one of the simplest primitives and we describe briefly how the processing is made for this case. Assuming that the RGB-D camera is calibrated and follows the pinhole model, the 3D point $\mathbf{P} \in \mathbb{R}^3$ corresponding to a pixel $\mathbf{p} = (u, v, 1)^T \in \mathbb{P}^2$ is:

$$\mathbf{P}(\mathbf{p}) = \mathbf{D}(\mathbf{p})\mathbf{K}^{-1}\mathbf{p}, \tag{2}$$

where $\mathbf{D}(\mathbf{p})$ is the depth image and $\mathbf{K} \in \mathbb{R}^{3\times 3}$ the intrinsic camera matrix. Using Equation 2 for all pixels inside the detected box creates a point cloud where the primitive model is fitted. For instance, the plane Π representing a "door" object has three degrees of freedom:

$$\Pi : \mathbf{n}^T\mathbf{P} + d = 0, \tag{3}$$

where $\mathbf{n}$ is the normal vector encoding the door orientation and the distance is $d = -\mathbf{n}^T\mathbf{P}_0$ for any point $\mathbf{P}_0 \in \Pi$.

For robustness and efficiency, the model parameters $(\mathbf{n}, d)$ are extracted using RANSAC [19], as well as to find all the pixels belonging to the model (i.e., all the door plane inliers).

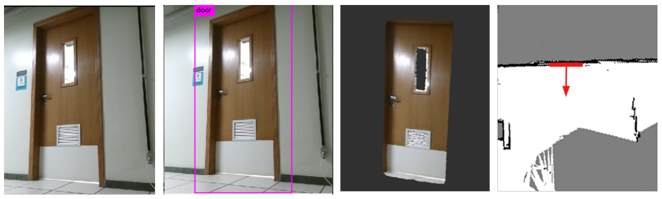

Fig. 3. Door detection: input image, object detection bounding box, RANSAC inliers for planar segmentation, object model represented in in the map.

Optimally, the image color information could also be jointly employed in the model fitting, we leave this consideration as future work. The position of the door $\mathbf{P}_d^{(c)}$ is then represented by the 2D projected centroid of the inlier points given by RANSAC. Finally, the object position $\mathbf{P}_d^{(c)}$ (2D projected centroid) and normal $\mathbf{n}$ are transformed from the camera frame to the global map frame $\mathbf{M}$, in order to build a unique map representation with the semantic classes:

$$\mathbf{P}_d = \mathbf{R}\mathbf{P}_d^{(c)} + \mathbf{t} \tag{4}$$

$$\mathbf{n}_d = \mathbf{R}\mathbf{n}, \tag{5}$$

where $(\mathbf{R}, \mathbf{t}) \in \mathbb{SE}(2)$ is the rotation and translation representing the current robot pose relative to the map frame. This is performed for each Yolo object observation denoted $\mathbf{y} = (\mathbf{P}_d, \arctan(\mathbf{n}))^T$. These steps can be seen in Figure 3.

C. *Object Position Tracking*

Another important concern is how to improve the information of the semantic classes in the map given multiple detections of the same object. It is worthy note that observations of the same object must consistently be associated with the same instance across different moments in time. Conversely, objects that have not been seen before must be instanced as new objects and stored in memory for future associations. In this section we present our solution to menage the objects instances viewed over time in the scene.

We first address the problem of deciding whether a current object detection have already been observed, or if it is a new instance. This is handled with a buffer of instances $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\}$ and then comparing each new detected object $\mathbf{y}$ with all the previously stored instances in the buffer. All stored instances in $\mathbf{X}$ that match the same semantic class of the observation are tested using the Mahalanobis distance:

$$r = \min_i \sqrt{(\mathbf{y} - \mathbf{x}_i)^T \mathbf{S}_i^{-1} (\mathbf{y} - \mathbf{x}_i)}, \tag{6}$$

where $\mathbf{x}_i$ is the i-th model of the n matched instances ($i = \{1, 2, 3, ..., n\}$) of $\mathbf{X}$ and $\mathbf{S}_i$ is its related covariance matrix. If the smallest distance r is less than δ, we assume it corresponds to a previously seen object; otherwise, a new object is instanced.

In order to combine different object observations, each stored semantic object is modeled temporally with a Kalman filter to maintain its state up-to-date and combine different observations. For each new observation of the object, the measured value is fed into the filter, and it will update the most probable object position and model (the *a posteriori* estate estimate) based on the previous estimation (the *a*

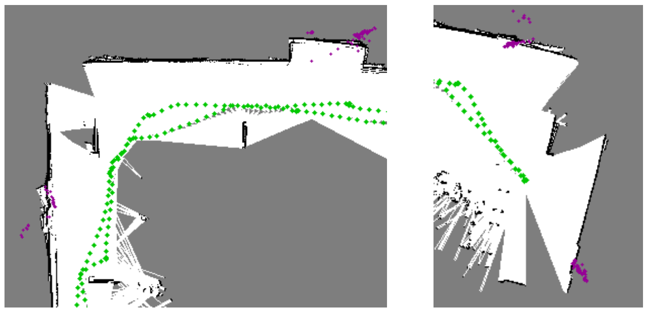
Fig. 4. Object observations during the robot navigation. The position of the objects observed over time are shown in pink.

priori estate estimate) and the new observation, as well as the covariance matrix associated with that estimation (a posteriori estimate covariance). Since objects of the class "door" are static, the estate transition and measurement dynamic models are simply the identity:

$$\left\{ \begin{array}{l} \mathbf{x}_i[k] = \mathbf{x}_i[k-1] + \tilde{\mathbf{w}}[k] \text{ and } \tilde{\mathbf{w}} \sim \mathcal{N}\left(\mathbf{0}_{3\times 1}, \mathbf{W}\right), \\ \mathbf{y}[k] = \mathbf{x}_i[k] + \tilde{\mathbf{z}}[k] \text{ and } \tilde{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{0}_{3\times 1}, \mathbf{Z}\right), \end{array} \right. \tag{7}$$

where the random variables $(\tilde{\mathbf{w}}, \tilde{\mathbf{z}})$ follow Normal distributions, $\mathbf{W}$ is the process covariance matrix and $\mathbf{Z}$ is the measurement covariance matrix.

This framework combines the information of the different observations as shown in Figure 4. The filter initialization and tunning details are described in Section V-A.

IV. DATASET

Apart from online experiments, we collected and tested our methodology on a dataset acquired in an indoor environment. The dataset contains raw sensor streams recorded from the robot using the rosbag toolkit. The dataset has a total of 15.2 GB and 5:03 min of duration (303 seconds). It is also provided with a bash file, `replay.sh` that runs the `rosbag play` command with the correct configurations used in our experiments (rate, simulation time, topic name remapping, static transformations, etc.). Another bash file, `slam-replay.sh`, has the same configurations, but also run the SLAM Gmapping algorithm using the parameters we set. This dataset includes RGB-D images, 2D LIDAR scans and odometry information whose details are:

- *RGB-D*: Both RGB and depth images have 640×480 resolution, $0.6m$ to $8.0m$ depth range and 60^o horizontal $\times$ 49.5^o vertical field of view. The camera used is the Orbec Astra at 10 Hz[2].
- *Laser Scans*: 180 degrees of scanning range with 0.36^o angular resolution, and $0.02m$ to $5.6m$ of depth range. The data were recorded using a Hokuyo URG-04LX-UG01 at 10 Hz.
- *Odometry*: This information is provided by the Kobuki base at 20 Hz. Traveled distance: 108.6m. Max speed: $0.57m/s$. Max angular: 1.07 rad/s. Covered Area: $42m \times 18.5m$.

[2]The original image frames were streamed at 30 Hz. We throttled this rate down to 10 Hz, with the purpose of shrinking the dataset size while maintaining its usability.

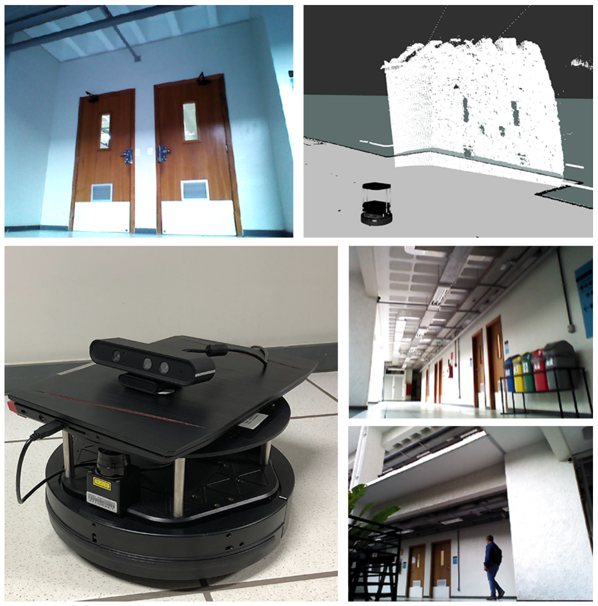
Fig. 5. Examples of scenes contained in the dataset and Kobuki base robot. The first row displays and RGB frame and its corresponding point cloud visualization. The robot with on-board sensors (RGB-D camera and 2D LIDAR) is shown in the bottom left image.

The footage contains different classes of objects: *people, window, door, bench, table, chair, trash bin, fire extinguisher*, as shown in the images of Figures 1 and 5. Every class considered static (i.e., all, except for people and chair) have their location specified in a ground truth map we provide. We also made available publicly this dataset and network configuration files used in the object detection module. This includes yolo-mapping.cfg, containing the network architecture; yolo-mapping.yaml containing the class names; and finally, yolo-mapping.weights, with the trained network weights used in our experiments.

V. EXPERIMENTS

In this section we evaluate our proposed framework using our indoor dataset. We start presenting the parameters tuning used in the experiments and then we show some semantic augmented map results.

A. System Setup and Implementation Aspects

Our platform consists of a Kobuki base robot with the same camera and laser sensors described in the dataset Section IV. The algorithm is implemented on ROS (Robot Operational System) and runs at 15 Hz in a laptop with Ubuntu 16.04, Intel core i7 and Nvidia GeForce 1050 Ti. Our goal was to initially detect and augment the map with relevant objects that do not usually change position over time, and one interesting candidate is to use doors. To this end, we trained the YOLO object detector with 150+ different images of doors.

The segmentation step was performed using the Point Cloud Library [20] implementation of RANSAC. We set optimize coefficients to true, and the distance threshold to 0.03. We use a value large enough to account for depth sensor measurement noise, but small enough that the door and the wall points lie in different planes. To perform object position tracking, we assume no correlation between the axis for the filter covariance matrices (i.e., the covariance matrices are diagonal matrices). We set the Kalman Filter initial covariance to $\sigma_{initial} = 3.0$; the measurement noise and process noise to $\mathbf{W} = 5 \times (\mathbf{I}_{3\times3})$, $\mathbf{Z} = 0.3 \times (\mathbf{I}_{3\times3})$, respectively. Finally, for the data association component, we set the Mahalanobis to varying values shown in Table I.

B. Augmented Mapping Results

The experiments were conducted in indoor scenes, containing large corridors with several doors. We tested with two different approaches for the localization module: one performing SLAM and another using pure localization on top of a previously generated map. We show some of the results for both of these approaches in Figures 6 and 7. The red dots represent the robot path, purple dots are unfiltered detected objects and the green lines are the filtered instances.

1) SLAM-based: The idea is to navigate in an unknown environment, while augmenting the map with semantic information. One issue we found with this approach is that during map generation of large spaces, bundle-adjustment and loop-closure algorithms will often displace fixed positions, causing some correctly placed objects to be misplaced. However, we found that this can be greatly improved by setting a threshold on the maximum distance of projected objects from the robot. Conversely, objects seen from far away (greater than six meters) will not be taking into account, which causes loss of possibly relevant information.

2) Localization-based: The SLAM module can easily be substituted by a pure localization algorithm, given the map and a valid starting position. This approach is useful for validating accuracy, as it is possible to compare projected objects positions to the marked ground truth. We tested this using different values for the Mahalanobis distance. The results can be seen in Table I. False positives represent the percentage of objects that were instantiated in excess, whilst false negatives represent the percentage of objects that were not instantiated.

TABLE I

RESULTS VARYING THE MAHALANOBIS DISTANCE (δ).

δ [m]	avg. error [m]	std [m]	false positives	false negatives
0.9	0.46	0.25	27.2%	0%
1.0	0.70	0.49	18.2%	0%
1.2	0.54	0.45	0.1%	0.1%
1.5	0.87	0.63	0%	0.1%

Smaller values for the distance threshold tend to cause a smaller error, but also favors the appearance of more false positives. That is because successive observations of the same door might not satisfy the distance condition. We observe that this is often the case when the objects are seen again while being revisited after the robot had traveled a longer distance in and out of the object's surrounding area. A reasonable explanation is that the localization module does not correct the accumulated robot pose error with enough accuracy, yielding larger deviations of the object's projection.

For bigger values the opposite is also true. In the case when two doors are relatively closer to each other, a bigger threshold will favor both of them to be interpreted as two different observations of the same object. One possible optimization to this hurdle would be to take into account the number of detected objects in the same frame, and accounting them as separate instances in the filtering stage.

We also found that even a small latency in the object detection stage (using the neural network) causes objects to often be projected in the wrong location in the 2D map while the robot moves, especially when it is turning. That is because the object is projected on the map using the *current* robot position and rotation, so the movement made by the robot during the detection stage causes the object to be positioned in a location different from that which the camera was facing when the image was captured.

The augmented maps obtained showed interesting results, as it can be observed in Figures 1, 6 and 7, although we observed practical limitations that could still be handled.

VI. CONCLUSIONS

We presented a methodology and an open framework for building augmented semantic maps. We also made available for the community a dataset containing 2D laser scans, RGB-D images and odometry. The sensor data is encapsulated in ROS bag files and is easy of use. The proposed methodology was tested both offline on the dataset and online, on a real robot. The framework was build on top of ROS, making use of the Yolo object detection system based on Convolutional Neural Networks and a SLAM implementation based on particle filter. We also utilized other open source libraries throughout the implementation, such as the OpenCV for both image processing and the Kalman filter module. The presented source code is highly modular, i.e., can be easily modified without the need to change other independent modules.

A possible extension to the presented work is increasing the number of classes in the detection step. This might require elaborating different and possibly more complex approaches for analyzing the point cloud information associated with that object. Other classes of objects such as furniture, appliances, and tools in general, especially organic objects such as people, animals, and plants would not follow these constraints and would therefore be harder to model. In the particular case of extending this approach to movable objects would imply altering the Kalman filter step, as the proposed matrices assumed static instances. Another interesting direction would be to improve the robot localization based on the semantics of the augmented map, which is seamless to how humans perform localization and navigation.

REFERENCES

[1] E. Rehder, F. Wirth, M. Lauer, and C. Stiller, "Pedestrian prediction by planning using deep neural networks," *CoRR*, 2018.

[2] R. V. Carneiro, R. C. Nascimento, R. Guidolini, V. B. Cardoso, T. Oliveira-Santos, C. Badue, and A. F. D. Souza, "Mapping road lanes using laser remission and deep neural networks," *CoRR*, vol. abs/1804.10662, 2018.

[3] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012.

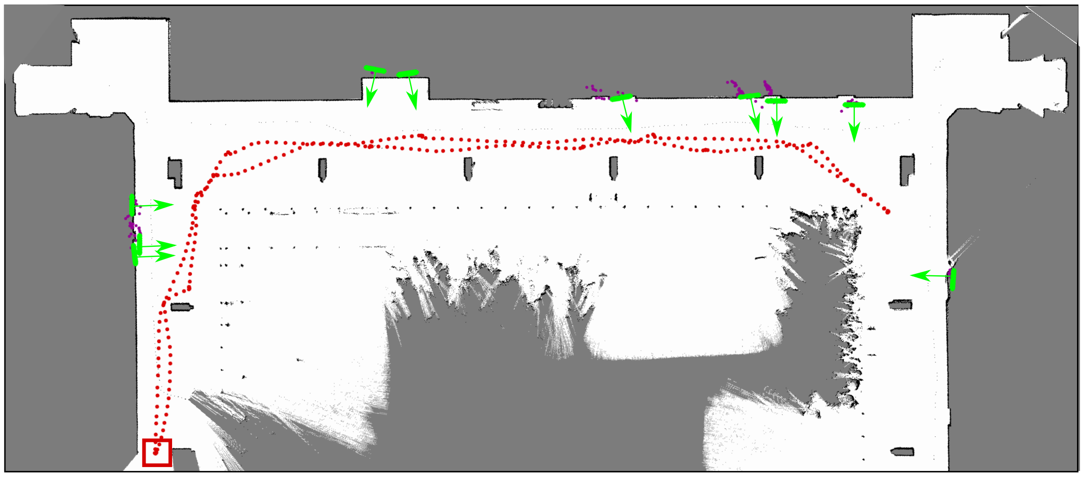

Fig. 6. Augmented 2D map with door instances with localization-only (figure axes dimensions $55.7 \times 24.3m$). The red square indicates the starting and ending point of the robot trajectory (red-dotted). Purple dots are the unfiltered positions observations and green lines are the doors filtered results. The reconstructed map depicts with green arrows the position and orientation of objects.

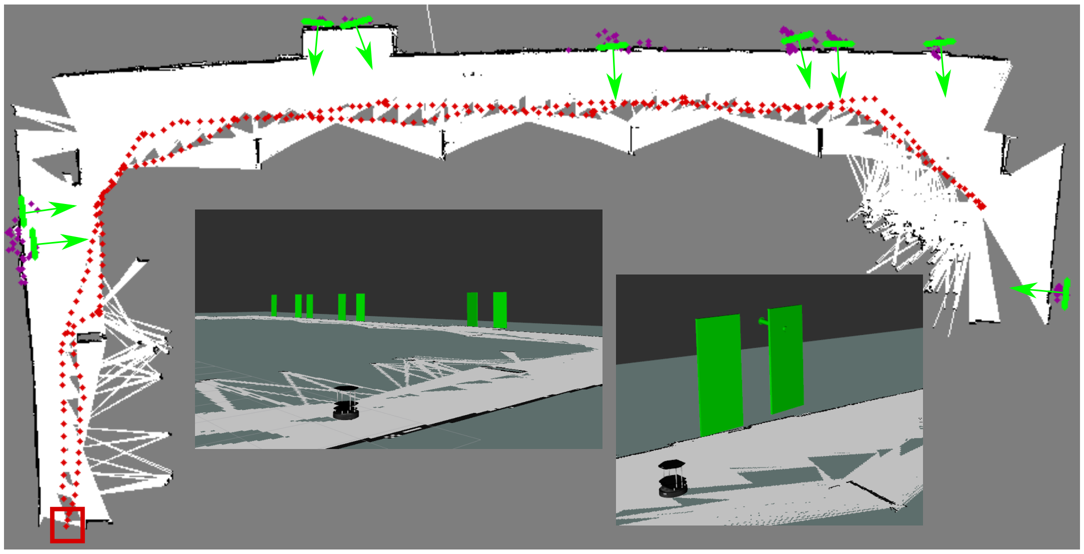

Fig. 7. Augmented 2D map with door instances using the SLAM localization system (figure axes dimensions $45.5 \times 22m$). Please see the legend described in Figure 6. The image depicts in green the position and orientation of the door objects, along with two visualizations of the augmented semantic map.

[4] P. Papadakis and P. Rives, "Binding human spatial interactions with mapping for enhanced mobility in dynamic environments," *Auton. Robots*, vol. 41, no. 5, pp. 1047–1059, 2017.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Pecognition*, 2016.

[9] E. R. Nascimento, G. L. Oliveira, M. F. M. Campos, and A. W. Vieira, "Improving Object Detection and Recognition for Semantic Mapping with an Extended Intensity and Shape based Descriptor," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Active Semantic Perception (ASP'11)*, San Francisco, CA, USA, Sept. 2011. [Online]. Available: http://activeperception.org/2011

[10] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *ICRA*. IEEE, 2017, pp. 4628–4635.

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

[12] X. Li and R. Belaroussi, "Semi-dense 3d semantic mapping from monocular slam," *arXiv preprint arXiv:1611.04144*, 2016.

[13] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *ECCV (2)*, ser. Lecture Notes in Computer Science, vol. 8690. Springer, 2014, pp. 834–849.

[14] "Intel realsense object library," https://software.intel.com/sites/products/realsense/object/developer_guide.html, 2017.

[15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[16] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[17] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Pecognition*, 2016, pp. 770–778.

[19] M. Zuliani, "Ransac for dummies," *Vision Research Lab, University of California, Santa Barbara*, 2009.

[20] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.