

Efficient Semantic Mapping in Dynamic Environments

Christian Hofmann^a, Mathias Fichtner^b, Markus Lieret^c and Jörg Franke^d

*Institute for Factory Automation and Production Systems, Friedrich-Alexander-Universität Erlangen-Nürnberg,
Egerlandstr. 7-9, 91058 Erlangen, Germany*

Keywords: UAV, Semantic Mapping, Object Detection, Information Fusion.

Abstract: Unmanned Aerial Vehicles (UAVs) are required to fulfill more and more complex tasks in indoor environments like inspection, stock-taking or transportation of goods. For these tasks, they need to percept objects and obstacles in the environment, navigate safely in it and sometimes even interact with it. A step towards generating a comprehensive environmental overview for robots are semantic maps. Nevertheless, UAVs have several constraints concerning size, weight and power consumption and thus computational resources. In this paper an efficient object-oriented semantic mapping approach suitable for UAVs and similar constrained robots in dynamic environments is proposed. The approach can be completely executed on a computer suited as onboard computer of an UAV. A map comprising semantic information and dynamic objects is generated and updated with update rate of more than 10 Hz.

1 INTRODUCTION

The number of applications using unmanned aerial vehicles (UAVs) is increasing rapidly. UAVs offer advantages for several indoor applications, for example for stock-taking (Kalinov et al., 2020) and transportation of goods (Lieret et al., 2019), as they can reach positions at a high altitude easily and move quickly. Especially in indoor environments, like in (Lieret et al., 2019) and (Kalinov et al., 2020), it is essential for an autonomous robot to have an up-to-date environment map, which is precise and rich of information about objects and obstacles in the environment, i.e. a semantic map. Semantic maps enable robots to fulfill complex tasks and improve their interaction with the environment, e.g., by improved path planning (Koch et al., 2019) and navigation.

There exist several approaches for semantic mapping (cf. section 2). Nevertheless, these approaches are not suited to be executed on resource constrained UAVs (or other similar constrained robots) while providing a sufficient map update rate to also include dynamic obstacles in the semantic map.

In this paper, we present a semantic mapping approach, which can run on hardware suited as an

UAV's onboard computer while providing a map update rate of more than 10 Hz for specific important objects, like persons or moving obstacles.

The main contributions of this paper are the architecture of the data processing pipeline for obtaining an object-oriented semantic map and the method of efficiently maintaining the object-oriented semantic map.

Due to the resource constraints on UAVs, it is not our goal to generate a semantic map comprising semantic information for all objects in the environment. Based on the discussion regarding semantic map representations in (Cadena et al., 2016), we have the opinion, that it is sufficient for many applications, if only semantic information of application relevant objects is incorporated in the map, while all other objects and obstacles are simply mapped as obstacles without further semantic information. Subsequently, our resulting map consists of two layers, a sparse semantic layer, comprising only highly relevant objects in the environment, and a dense obstacle map layer, comprising no semantic information.

The paper is structured as follows: First, we present and discuss related work and motivate our mapping approach. In the next two sections, our data processing pipeline and the semantic mapping method is introduced. In section 5 experiments and the evaluation in a real world scenario are presented. Finally, a brief summary and outlook is given.

^a  <https://orcid.org/0000-0003-1720-6948>

^b  <https://orcid.org/0000-0001-9335-4884>

^c  <https://orcid.org/0000-0001-9585-0128>

^d  <https://orcid.org/0000-0003-0700-2028>

2 RELATED WORK

There exist several approaches for building and updating semantic maps. Sunderhauf et al. propose a semantic mapping approach, that creates an object-oriented semantic map by using a convolutional neural network (CNN) for object detection in camera images followed by a plane based 3D segmentation to obtain the objects' 3D position and shape (Sunderhauf et al., 2017). The data association for updating the map with newly detected objects is based on their euclidean distance and point cloud similarity. Another object-oriented approach is presented in (Nakajima and Saito, 2019). It is running in real-time, but on powerful hardware, that is not suited to be mounted on an UAV. The algorithm uses a combination of a CNN for object detection in images and geometric segmentation of 3D data to generate 3D object detections for the semantic map. In (Grinvald et al., 2019) a volumetric, i.e. dense, semantic map is created by an instance segmentation neural network and geometric segmentation based on RGB-D camera data to detect and segment objects in 3D. The data association is mainly based on matching 3D data. This approach needs high computational resources and runs with a low update rate. A semantic simultaneous localization and mapping (SLAM) approach is presented in (McCormac et al., 2018). It uses an instance segmentation CNN for detecting objects in camera images. The semantic mapping is tightly integrated in the SLAM process.

An approach for semantic mapping and path planning for UAVs is presented in (Koch et al., 2019). The approach targets 3D-Reconstruction using images captured by an UAV in outdoor scenarios. For generating a semantic map, first images of the area are captured during an exploration flight. With these images a semantic reconstruction of the environment is generated using pixel-wise dense semantic segmentation by a fully convolutional network and structure from motion and multi-view stereo pipelines using an offboard computer. Based on the semantic environment reconstruction, an improved flight path for image acquisition for the 3D reconstruction is generated. In (Maturana et al., 2017) a 2.5 D semantic mapping approach using the UAV's onboard computer for outdoor scenarios is presented. With a novel semantic segmentation neural network objects are segmented in camera images. By ray casting, the object detections are projected on a digital elevation map of the environment to generate a 2.5 D semantic grid map.

None of these approaches is suited for indoor semantic mapping with autonomous UAVs:

Since dynamic objects like persons are likely present in indoor environments, a sufficient map update rate is necessary for navigation based on the semantic map. The semantic mapping approaches in (Sunderhauf et al., 2017), (Nakajima and Saito, 2019) (Grinvald et al., 2019) and (McCormac et al., 2018) are not capable of providing such an update rate on currently available computers suited for onboard use on UAVs. Additionally, including dynamic objects in the map is not addressed by these approaches.

As performed for generating the semantic environment reconstruction by (Koch et al., 2019), computational demanding tasks could be outsourced to an offboard computer. However, thus the UAV would not be autonomous. Our approach is completely executable on an UAV's onboard computer. Further, the approaches in (Koch et al., 2019) and (Maturana et al., 2017) are designed for outdoor scenarios. For indoor navigation, a 3D map of all obstacles is necessary, while in our opinion only semantic information of relevant objects in the environment is sufficient. Nevertheless, the objects in the semantic map should be also mapped in 3D.

We also do not target a semantic SLAM system, like presented in (McCormac et al., 2018). Commonly, the pose and state of the UAV is estimated by a flight control unit (FCU) on the UAV. Thus, the UAV's pose is already available. In indoor scenarios the FCU's pose estimation can be enhanced e.g., by indoor radio based localization (Lieret et al., 2019) or visual odometry (e.g., Intel Tracking Camera T265).

3 DATA PROCESSING PIPELINE

An overview of our proposed data processing pipeline for obtaining the object-oriented semantic map is given in Figure 1. We define consecutive modules for data processing. Our approach relies on color images and 3D point clouds aligned with the images as input data. This sensory information is either directly emitted or can be computed by using a RGB-D camera. Cameras of this type are lightweight and small while providing a sufficiently large field-of-view, making them suited for UAVs. We assume the pose of the UAV and thus of the camera is provided.

3.1 Object Detection

Our approach is designed to work with one or multiple object detectors using the color images as input. Considering computational efficiency, the use of a single object detector is preferable. Nevertheless,

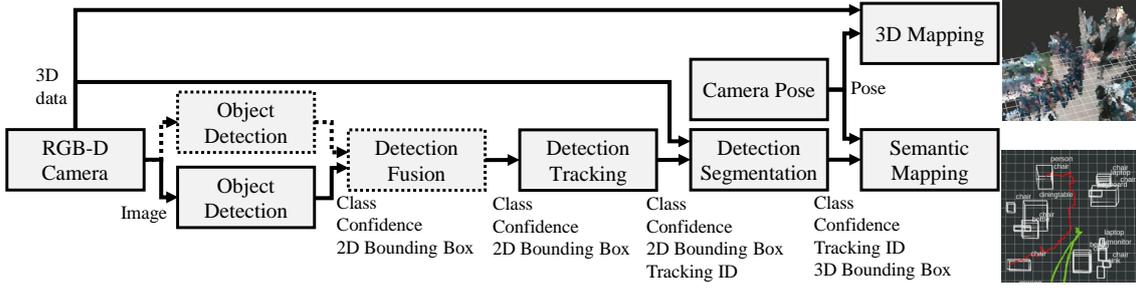


Figure 1: Overview of our proposed data processing pipeline for object-oriented semantic mapping in dynamic environments.

it has been shown that fusing multiple object detectors can improve the detection results (Solovyev et al., 2021) (Hofmann et al., 2019). The pipeline is modular, so that, depending on the application and the available computational resources, both options can be implemented without any adaptations in the pipeline. In case of using multiple detectors, they are intended to perform object detection simultaneously.

Each detection $\mathbf{d}_i \in \mathcal{D}_t$, with \mathcal{D}_t denoting all detections provided by a detector in an image captured at time t and $i \in [0, I]$, where I is the total number of detections in \mathcal{D}_t , should comprise following elements:

- Class s_i
- Confidence score c_i for the detection, $c_i \in [0, 1]$
- 2D bounding box \mathbf{b}_i^{2D} enclosing the detection

Since it is relevant for updating the semantic map, the detectors must also provide an output if no detections are made by passing the timestamp t of the image processed to semantic mapping module. Consequently, t must be also passed through the following modules.

For the evaluation in section 5 we have implemented the pipeline using these exemplary detectors:

- The CNN Tiny-YOLOv4 (following named YOLO) (Bochkovskiy et al., 2020)
- The approach for detecting moving objects with a moving camera presented in (Yi et al., 2013)

These two detectors were chosen for our particular application since they cover two essential kinds of object types: 1) objects highly relevant to the robot’s task which are known a priori and can thus be learnt (YOLO, in section 5 weights based on the COCO dataset are used), and 2) dynamic obstacles, i.e. moving objects. Nevertheless, any object detector that provides bounding boxes as detection result could be integrated into the pipeline.

3.2 Detection Fusion

In case of using multiple detectors, following all detections \mathcal{D}_t provided by the detectors originating in the same image are fused. As shown for example in

(Solovyev et al., 2021) and (Hofmann et al., 2019) the bounding boxes, classes and confidences are sufficient to fuse the detections.

For the implementation of the pipeline to evaluate the approach (section 5), we have adapted the algorithm presented in (Solovyev et al., 2021) for our detector selection: The incoming detections can consist of detections by YOLO $\mathcal{D}_t^Y \subseteq \mathcal{D}_t$ and motion detections $\mathcal{D}_t^{MO} \subseteq \mathcal{D}_t$. First, for each YOLO detection $\mathbf{d}_j^Y \in \mathcal{D}_t^Y$, with $j \in [0, J]$ and J being the total number of detections in \mathcal{D}_t^Y , the intersection-over-union (IOU) with each motion detection $\mathbf{d}_k^{MO} \in \mathcal{D}_t^{MO}$, with $k \in [0, K]$ and K being the total number of detections in \mathcal{D}_t^{MO} , using their respective bounding boxes $\mathbf{b}_j^{2D,Y}$ and $\mathbf{b}_k^{2D,MO}$ is calculated. Following the detections are clustered by assigning each motion detection $\mathbf{d}_k^{MO} \in \mathcal{D}_t^{MO}$ to the YOLO detection $\mathbf{d}_j^Y \in \mathcal{D}_t^Y$ with which it has the highest overall IOU, but only if this IOU exceeds a minimal threshold. Subsequently, if there is more than one bounding box in a cluster, i.e. one or more \mathbf{d}_k^{MO} are assigned to \mathbf{d}_j^Y , the bounding boxes are fused. The bounding box $\mathbf{b}_j^{2D,Fusion}$ and the confidence score c_j^{Fusion} of the resulting detection \mathbf{d}_j^{Fusion} is calculated based on the original bounding boxes, as described in (Solovyev et al., 2021). With our exemplary detectors, the semantic information is fused according to (Hofmann et al., 2019), resulting for example in the class dynamic person, if a detected person is fused with a motion detection.

The output of this step are the fused detections \mathcal{D}_t^{Fusion} and all detections that were not fused, all together following again denoted by \mathcal{D}_t .

3.3 Detection Tracking

We integrated object tracking in our pipeline, because the data association between object detections in consecutive images can thus already be performed computationally efficient in the 2D image space. Tracking based on bounding boxes is demonstrated for example in (Bewley et al., 2016). In this processing step, a tracking identifier (ID) a is added to all input detec-

tions \mathcal{D}_t . If a detection $\mathbf{d}_i \in \mathcal{D}_t$ is tracked, i.e. it is associated with a prior detection, its tracking ID a_i is set to a specific value, which is the same for all detections associated with each other by the same tracker (Bewley et al., 2016). If a detection \mathbf{d}_i can't be tracked, the tracking ID should be set to a defined value. All detections \mathcal{D}_t are passed to the following segmentation module. The algorithm described in (Bewley et al., 2016) is used for the evaluation in Section 5.

3.4 Detection Segmentation

Since the goal is a 3D object-oriented semantic map, the detected objects are transformed from image space to 3D space and segmented.

For each incoming detection $\mathbf{d}_i \in \mathcal{D}_t$, the 3D points $\mathcal{P}_i \subseteq \mathcal{P}_t$ corresponding to the pixels inside the detection's 2D bounding box \mathbf{b}_i^{2D} are extracted from the point cloud \mathcal{P}_t . The point cloud \mathcal{P}_i corresponds to the color image in which the detections \mathcal{D}_t were made. Since often the bounding box \mathbf{b}_i^{2D} encloses not solely the detected object, but also pixels and subsequently 3D points belonging to the background and other objects, each detection's point cloud is filtered. First, the point cloud is downsampled using a voxel grid filter. Following, the point cloud \mathcal{P}_i is clustered using euclidean cluster extraction. For determining, which cluster $\mathcal{L}_r \subseteq \mathcal{P}_i$, with $r \in [1, R]$ and R being the total number of clusters found in \mathcal{P}_i , represents the object detected in the image, we rely on two characteristics: 1) Normally, the bounding box \mathbf{b}_i^{2D} should enclose the detected object tightly in the image, so that most pixels in \mathbf{b}_i^{2D} belong to the detected object and thus most points in \mathcal{P}_i . 2) Further, the detected object should most probably be present in the center of the bounding box \mathbf{b}_i^{2D} (Hofmann et al., 2019). The cluster $\mathcal{L}_r \subseteq \mathcal{P}_i$ is chosen to represent the detected object, which has the maximal score

$$v_{\mathcal{L}_r} = w \cdot \frac{q_{\mathcal{L}_r}}{q_{\mathcal{P}_i}} + (1 - w) \cdot \frac{h_{\mathcal{L}_r}}{h_{\mathcal{P}_i, \max}}, \quad (1)$$

where $v_{\mathcal{L}_r}$ is the resulting score of the cluster \mathcal{L}_r , $w \in [0, 1]$ is a weighting factor for the influence of the two characteristics, $q_{\mathcal{L}_r}$ is the number of points in the cluster \mathcal{L}_r and $q_{\mathcal{P}_i}$ is the number of points in \mathcal{P}_i . $h_{\mathcal{L}_r}$ is the euclidean distance of the cluster's center to the 3D center of the 2D bounding box \mathbf{b}_i^{2D} . This 3D center is prior calculated by extracting the points of \mathcal{P}_i corresponding to the pixel in a certain area around the 2D center of \mathbf{b}_i^{2D} and calculating their mean 3D position. $h_{\mathcal{P}_i, \max}$ is the maximal possible euclidean distance between two points in \mathcal{P}_i , i.e. the euclidean distance of an imaginary point $\mathbf{x}_{\min} = [x_{\min}, y_{\min}, z_{\min}]^T$, where $[x_{\min}, y_{\min}, z_{\min}]^T$ are the minimal coordinates

for each axis present in the point cloud \mathcal{P}_i , to an imaginary point \mathbf{x}_{\max} with the maximal coordinates $[x_{\max}, y_{\max}, z_{\max}]^T$ for each axis in \mathcal{P}_i .

The detected object's 3D position $\mathbf{x}_i^{\text{camera}}$ is defined as the center of the cluster \mathcal{L}_r with the highest score $v_{\mathcal{L}_r}$. The 3D bounding box $\mathbf{b}_i^{3D, \text{camera}}$ is calculated to enclose all points in this cluster. The position $\mathbf{x}_i^{\text{camera}}$ and 3D bounding box $\mathbf{b}_i^{3D, \text{camera}}$ are at this step expressed in relation to the camera frame, not the map frame, which differ as the camera is moving.

In case the calculation of $\mathbf{x}_i^{\text{camera}}$ and $\mathbf{b}_i^{3D, \text{camera}}$ is not possible, e.g., if there are no valid points in the point cloud, the detection \mathbf{d}_i is not passed further. If this is the case for all $\mathbf{d}_i \in \mathcal{D}_t$, the timestamp t must be passed for updating the map, providing the information that no valid detections were made.

We use 3D bounding boxes to represent objects in the map, as they are computational more efficient than point clouds. However, also the segmented point clouds could be used for the semantic mapping.

We developed this detector-independent segmentation approach, since by this the detectors can be easily changed for different applications, while the segmentation can be always applied. The implementation for the evaluation in section 5 is based on the point cloud library (PCL) (Rusu and Cousins, 2011).

3.5 Mapping

Based on the results of the segmentation step, the sparse object-oriented semantic map is created. To also map objects and obstacles that cannot be detected, a further mapping approach should be used to create a complete (non-semantic) 3D map of the environment (cf. Figure 1). For example, RTAB-Map (Labbé and Michaud, 2019), which we also use in the experiments in section 5, is well suited, as it also works with RGB-D data. Simultaneous to the map generated by RTAB-Map, our semantic mapping approach outputs the object map comprising relevant and moving objects (depending on the detector selection) at a relatively high update rate.

4 SEMANTIC MAPPING

Our object-oriented semantic map at time t

$$\mathcal{M}_t = \{\mathbf{o}_{1,t}, \dots, \mathbf{o}_{N,t}\} \quad (2)$$

consists of N objects $\mathbf{o}_{n,t}$ with $n \in [0, N]$. The map \mathcal{M}_t is time dependent, as objects can be added or deleted, depending on the detections \mathcal{D}_t made in the image captured at time t . Following the time (i.e. the times-

tamp) corresponding to the detections made right before the current detections \mathcal{D}_t is denoted as $t - 1$. This means \mathcal{M}_{t-1} denotes the map before updating it with the detections \mathcal{D}_t , resulting in \mathcal{M}_t .

Each object $\mathbf{o}_{n,t} \in \mathcal{M}_t$ comprises the following information:

- Object class $s_{n,t}$
- Indicator $g_{n,t}$, whether the object is currently dynamic ($g_{n,t} = \text{true}$) or static ($g_{n,t} = \text{false}$),
- Tracking ID $a_{n,t}$, which is the tracking ID of the last detection associated with the object
- The object's position $\mathbf{x}_{n,t}^{\text{map}}$ in the map
- 3D Bounding Box $\mathbf{b}_{n,t}^{\text{3D,map}}$ to store the object's size
- Counter for the number of updates of $\mathbf{o}_{n,t}$
- Persistence filter (Rosen et al., 2016), which estimates whether an object is still present in the environment or disappeared
- Persistence probability $u_{n,t}$, which indicates the probability, that the object is still present, estimated by the object's persistence filter

Besides the current detections \mathcal{D}_t , the camera's pose $\mathbf{p}_t^{\text{map}}$ at time t in the map is passed to the mapping module (cf. Figure 1). The map \mathcal{M}_{t-1} is updated to \mathcal{M}_t with every incoming object measurement, even if no detections (so only the timestamp t of the input image) is passed. The update step to generate \mathcal{M}_t from \mathcal{M}_{t-1} is structured as follows:

1) Detection Transform Step: For each incoming detection $\mathbf{d}_i \in \mathcal{D}_t$, its position $\mathbf{x}_i^{\text{camera}}$ and 3D bounding box $\mathbf{b}_i^{\text{3D,camera}}$ are transformed from the camera frame to the map frame by using the camera's pose $\mathbf{p}_t^{\text{map}}$ resulting in $\mathbf{x}_i^{\text{map}}$ and $\mathbf{b}_i^{\text{3D,map}}$.

2) Expectation Step: With $\mathbf{p}_t^{\text{map}}$, it is calculated, which objects in the prior map, $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$, should be detected in current image and so be in \mathcal{D}_t by projecting each object's 3D position $\mathbf{x}_{n,t-1}^{\text{map}}$ that is in a certain radius in front of the camera, into the camera's image plane. For this, the camera's intrinsic parameters are necessary and must be provided once when starting the mapping. All objects $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$ that should be visible in the current image and so be in \mathcal{D}_t are marked as "expected objects".

3) Data Association Step: First, it is checked for each incoming detection $\mathbf{d}_i \in \mathcal{D}_t$, if its tracking ID a_i is already stored in the tracking ID $a_{n,t-1}$ of any object $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$. If so, it is checked if the euclidean distance between the detection's position $\mathbf{x}_i^{\text{map}}$ and object's position $\mathbf{x}_{n,t-1}^{\text{map}}$ is smaller than a threshold distance. If this condition is fulfilled, the detection \mathbf{d}_i is associated with the object $\mathbf{o}_{n,t-1}$. Otherwise, each

detection's position $\mathbf{x}_i^{\text{map}}$ and class s_i with all not yet associated objects $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$ are compared. It is checked whether the euclidean distance between the positions $\mathbf{x}_i^{\text{map}}$ and $\mathbf{x}_{n,t-1}^{\text{map}}$ is smaller than a threshold distance and the classes s_i and $s_{n,t-1}$ are equal. In case these conditions are fulfilled, the IOU between the detection's 3D bounding box $\mathbf{b}_{3D,i}^{\text{map}}$ and the object's 3D bounding box $\mathbf{b}_{3D,n,t-1}^{\text{map}}$ is calculated. For this, the detection's 3D bounding box $\mathbf{b}_{3D,i}^{\text{map}}$ is shifted to the object's position $\mathbf{x}_{n,t-1}^{\text{map}}$. If the 3D IOU value exceeds a threshold, \mathbf{d}_i is noted as association candidate for $\mathbf{o}_{n,t-1}$. The detection \mathbf{d}_i is associated with the object $\mathbf{o}_{n,t-1}$ providing the highest IOU. The underlying assumption is, that a real world object's size and so the resulting 3D bounding box should be relatively constant. This process is inspired by the data association described in (Sunderhauf et al., 2017).

4) Map Update Step: If an object $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$ is associated with a detection $\mathbf{d}_i \in \mathcal{D}_t$, following update steps to obtain $\mathbf{o}_{n,t}$ are performed:

The class $s_{n,t}$ of $\mathbf{o}_{n,t}$ is set to the detection's class s_i . However, $s_{n,t-1}$ and s_i and consequently $s_{n,t}$ should be identically based on the data association.

If \mathbf{d}_i is a dynamic object, $g_{n,t}$ is set true, otherwise it is set to false.

If the detection \mathbf{d}_i has a valid tracking ID a_i , the object's tracking ID $a_{n,t}$ is updated with this one.

If the object is dynamic, i.e. $g_{n,t} = \text{true}$, the object's new position $\mathbf{x}_{n,t}^{\text{map}}$ is set to the detection's position $\mathbf{x}_i^{\text{map}}$, assuming the object is moving. Else, we assume the object is static, $\mathbf{x}_{n,t}^{\text{map}}$ is calculated as the weighted mean of $\mathbf{x}_i^{\text{map}}$ and $\mathbf{x}_{n,t-1}^{\text{map}}$, where $\mathbf{x}_i^{\text{map}}$ has the weight 1 and $\mathbf{x}_{n,t-1}^{\text{map}}$ has a weight equal to the number of the object's prior updates.

The weighted mean is also used in the same manner to update the size of the object's 3D bounding box $\mathbf{b}_{n,t}^{\text{3D,map}}$ based on $\mathbf{b}_i^{\text{3D,map}}$ (weight 1) and $\mathbf{b}_{n,t-1}^{\text{3D,map}}$ (weight equal to the number prior updates). The center of $\mathbf{b}_{n,t}^{\text{3D,map}}$ is $\mathbf{x}_{n,t}^{\text{map}}$.

The number of updates for $\mathbf{o}_{n,t}$ is increased by 1.

If an object $\mathbf{o}_{n,t-1}$ is marked as "expected object", but is not associated with any current detection $\mathbf{d}_i \in \mathcal{D}_t$, the object's persistence filter is accordingly updated with "false", i.e. the object was not detected, leading to a decreased persistence probability $u_{n,t}$ of $\mathbf{o}_{n,t}$.

If an object $\mathbf{o}_{n,t-1}$ is marked as "expected object", and is associated with a current detection $\mathbf{d}_i \in \mathcal{D}_t$ the object's persistence filter is accordingly updated with "true", i.e. the object was detected, leading to a increased or static persistence probability $u_{n,t}$.

Not expected static objects $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$, that are not associated with a detection $\mathbf{d}_i \in \mathcal{D}_t$, are not updated.

For not expected dynamic objects $\mathbf{o}_{n,t-1} \in \mathcal{M}_{t-1}$,

which are not associated with a detection, the persistence probability $u_{n,t}$ for the current time t is predicted using its persistence filter, normally leading to a decreased persistence probability $u_{n,t}$ of $\mathbf{o}_{n,t}$. This is based on the assumption that the object moves, i.e. it should also disappear in the map.

If a detection $\mathbf{d}_i \in \mathcal{D}_t$ is not associated with any object in \mathcal{M}_{t-1} , it is added as new object to the map \mathcal{M}_t .

Finally, after this update process it is checked, whether the persistence probability $u_{n,t}$ of any object $\mathbf{o}_{n,t} \in \mathcal{M}_t$ is lower than a threshold. If so, $\mathbf{o}_{n,t}$ is deleted from map, assuming it disappeared. All remaining objects $\mathbf{o}_{n,t}$ are then representing the updated object-oriented semantic map \mathcal{M}_t .

5 EXPERIMENTS AND EVALUATION

Following, the evaluation of our approach in a real world scenario with focus on the semantic map is presented. We use an Intel Realsense D435 RGB-D camera with the resolution set to 640 x 480 pixel (RGB and aligned point cloud) and the frame rate set to 15 Hz. To continuously obtain the 3D pose of the RGB-D camera, we use an Intel Realsense T265 Tracking camera, fixed with the RGB-D camera on a plate (total size: 108 x 100 x 30 mm). As computer for data processing we use an NVIDIA Jetson Xavier NX (15 watt and 6 cores mode), which is well suited to be used on UAVs (size with case for mounting: 130 x 92 x 41 mm). The setup (computer with case, two cameras with plate) has a total weight of 541 grams. Due to safety reasons, we carried the Jetson and the cameras by hand for data acquisition. We saved the raw camera data and used it to evaluate our approach with different configurations running on the Jetson.

Only detections by YOLO, which is executed using the Jetson's GPU, that have a detection confidence exceeding 0.5 are passed to the pipeline. Further, detections by the motion detector are filtered by only passing detections enclosing more than 1000 pixels to reduce false detections caused by camera motion. For the detection segmentation, we choose the down-sampling voxel size to 0.05 m and the weight factor to $w = 0.75$. Further, only segmented detections that are within a radius of 5 m around the camera are used, as the point cloud provided by the D435 camera is very noisy in larger distances. For the persistence filter we set, as suggested by (Rosen et al., 2016), exponential priors for the survival function with the fixed rate parameters of 0.001 in case of static objects and 1.5 otherwise. By this, dynamic objects are deleted relatively fast if they are not detected any more, while

static objects remain in the map. We set the probability for false detections to 0.3, for missed detections to 0.6, empirically based on the used detectors and the scenario. For data association, we set the distance threshold to 1.0 m and the minimal IOU to 0.01.

Using the two detectors, a semantic map update rate of 10 to 15 Hz on the Jetson is obtained, depending on the observed scene and the number of objects in the semantic map. This means, not every camera measurement is incorporated in the semantic map. All prior modules of our pipeline including the segmentation module output their results at a rate of 15 Hz, thus, they evaluate almost all input data. The Jetson's CPU is utilized between 60 and 90 percent when executing the whole pipeline comprising the two detectors as well as RTAB-Map in SLAM-Mode. This value is reduced to 40 to 70 percent while the semantic map update rate is increased to 14 to 15 Hz, when using YOLO as only detector, but no motion detection and consequently no detection fusion. Using this configuration, it is possible to execute further computationally intensive tasks onboard. Further software optimizations should also reduce the CPU utilization.

Following, the key findings obtained by our experiments are described. Figure 2a shows the point cloud map obtained by RTAB-Map without any semantic information. The point cloud map of RTAB-Map is not cleared, thus the person moving (from the middle left in the picture to the top) appears several times, displaying the trajectory well.

In Figure 2b the semantic map generated using the two object detectors is pictured. The detected semantic objects fit well to the point cloud map. Nevertheless, there are several false objects in the map as well as missing objects. This is mainly based on wrong and missed detections by YOLO. Thus, a more accurate detector should improve the result. However, often the trade-off between detection accuracy and computational efficiency has to be considered. Another influence to the object detection are blurry images output by the D435 camera due to camera motion. This makes it difficult to detect objects and causes wrong motion detections (e.g., a tv-monitor with a red box in the bottom right). A camera with global shutter should solve this problem.

Further, it is not easy to determine whether some detections are false or correct. An overview is given in Figure 2b by the check marks for correct objects in the map and x's for wrong objects. In several cases YOLO detected objects with similar classes in different images for the same real world object. Examples are a chair also detected as bench in the middle right and several monitors also detected as laptop. To overcome this, a data association algorithm that also com-

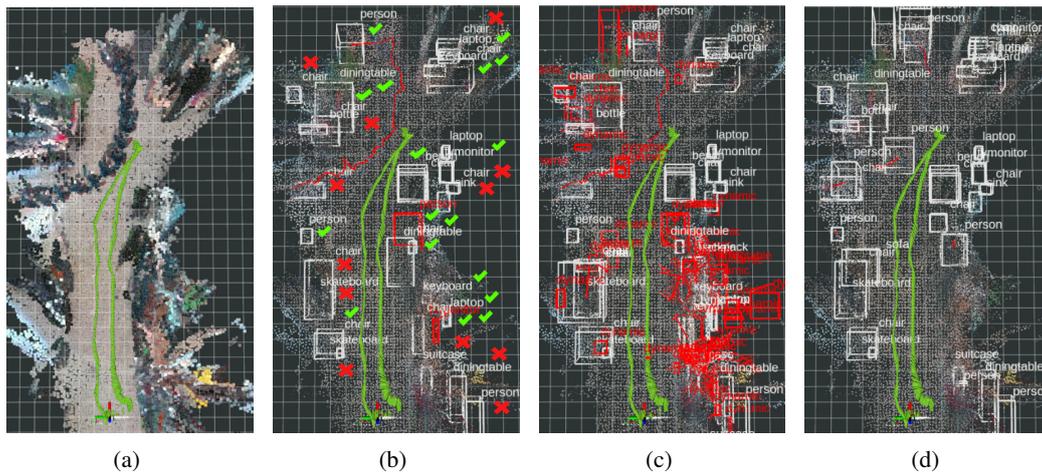


Figure 2: Resulting maps of our real world scenario comprising the point cloud map generated by RTAB-Map and our object-oriented semantic map with three different configurations. Picture (a) shows the point cloud map of RTAB-Map without our semantic map and point size set to 0.1 m. Pictures (b), (c), and (d) present maps obtained using the configuration described in section 5 with two detectors (b), with deactivated persistence filter for all objects (c) and with YOLO as only object detector (d). The point size of the point cloud map generated by RTAB-Map in those maps is set to 0.03 m. Static objects are displayed by white bounding boxes, dynamic ones with red. The trajectory of all persons is pictured as red line. Green arrows show the camera’s trajectory. The green check marks in (b) show objects in the map that were correctly mapped, the red x’s show false mapped objects. The grid has a cell size of 0.5 m.

prises the objects’ semantics could be used. Summed up, there are 17 correct objects and 12 false objects in the map. Nevertheless, all persons as most safety relevant objects were detected correctly. Several objects present in the real scenario are missing in the map, because they were not detected by YOLO. Again, a more accurate object detector explicitly trained for the environment should improve the result.

The detected objects’ sizes fit mostly well to the point cloud and the real world objects. Though, objects that are surrounded by many other objects have often a too big 3D bounding box based on the point cloud segmentation. As displayed by the camera trajectory in Figure 2, many objects were mapped while the camera only passed them. If an object is well visible in the image and the point cloud, like the person at the top of the map, it is well detected and segmented. Comparing pictures 2a and 2b, it is also obvious that the trajectory of the moving person is correctly mapped.

Picture 2c shows a map with the each object’s persistence filter deactivated, thus no objects are deleted from the map. Accordingly, there are many wrong motion detections in the map, originating from camera movement. Comparing this with Figure 2b, all wrong dynamic objects are filtered from the map using persistence filters for each object. Summarized, the persistence filter works as intended and is a valuable object attribute to maintain the semantic map.

Figure 2d shows the map obtained when using YOLO as only object detector. Comparing the seman-

tic map to the one in Figure 2b, the result is similar. However, the moving person was not always detected by YOLO and thus associated, resulting in several mapped static persons. The motion detector detected the moving person in some images (cf. Figure 2b), in which YOLO could not detect the person. Thus, the simultaneous motion detection is advantageous, but probably not necessary for all applications. Whether an object is probably dynamic or static, could for example also be inferred from the object class.

Further differences between the maps in the Figures 2b and 2d are several missing objects at the bottom right in Figure 2d. These differences occur, because with YOLO as only detector, more measurements are incorporated in the semantic map (higher map update rate). Thus, the missing objects are deleted due to a low persistence probability caused by missed detections that were now incorporated in the map.

6 CONCLUSIONS AND FUTURE WORK

We presented an approach for object-oriented semantic mapping in dynamic environments, which is suited for UAVs and similar robots constrained concerning size, weight and power consumption and thus computational resources. When using only one object detector (in our experiments Tiny YOLO), the approach is

sufficiently computationally efficient to allow the execution of further software on the onboard computer. Software optimizations should allow the use of multiple detectors, leading to an improved mapping result.

Our approach achieves satisfying results in the real world evaluation. Nevertheless, the data association for semantic mapping could be improved by using semantic information, e.g., presented in (Doherty et al., 2020). In addition, more object attributes, like uncertainties (Hiller et al., 2018), could be integrated. Further, we plan to investigate our approach using object detectors with higher accuracy.

ACKNOWLEDGEMENTS

This work was developed within the project "AIRKom" funded by the Forschungsgemeinschaft Intralogistik / Foerdertechnik und Logistiksysteme (IFL) e.V.

REFERENCES

- Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (25.09.2016 - 28.09.2016). Simple online and real-time tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE.
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (23.04.2020). Yolov4: Optimal speed and accuracy of object detection.
- Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.
- Doherty, K. J., Baxter, D. P., Schneeweiss, E., and Leonard, J. J. (31.05.2020 - 31.08.2020). Probabilistic data association via mixture models for robust semantic slam. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1098–1104. IEEE.
- Grinvald, M., Furrer, F., Novkovic, T., Chung, J. J., Cadena, C., Siegwart, R., and Nieto, J. (2019). Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044.
- Hiller, M., Particke, F., Hofmann, C., Bey, H., and Thielecke, J. (19.07.2018 - 23.07.2018). World modeling for mobile platforms using a contextual object-based representation of the environment. In *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 187–191. IEEE.
- Hofmann, C., Particke, F., Hiller, M., and Thielecke, J. (25.02.2019 - 27.02.2019). Object detection, classification and localization by infrastructural stereo cameras. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 808–815. SCITEPRESS - Science and Technology Publications.
- Kalinov, I., Petrovsky, A., Ilin, V., Pristanskiy, E., Kurenkov, M., Ramzhaev, V., Idrisov, I., and Tsetserukou, D. (2020). Warevision: Cnn barcode detection-based uav trajectory optimization for autonomous warehouse stocktaking. *IEEE Robotics and Automation Letters*, 5(4):6647–6653.
- Koch, T., Körner, M., and Fraundorfer, F. (2019). Automatic and semantically-aware 3d uav flight planning for image-based 3d reconstruction. *Remote Sensing*, 11(13):1550.
- Labbé, M. and Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446.
- Lieret, M., Kogan, V., Doll, S., and Franke, J. (22.08.2019 - 26.08.2019). Automated in-house transportation of small load carriers with autonomous unmanned aerial vehicles. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1010–1015. IEEE.
- Maturana, D., Arora, S., and Scherer, S. (24.09.2017 - 28.09.2017). Looking forward: A semantic mapping system for scouting with micro-aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6691–6698. IEEE.
- Mccormac, J., Clark, R., Bloesch, M., Davison, A., and Leutenegger, S. (05.09.2018 - 08.09.2018). Fusion++: Volumetric object-level slam. In *2018 International Conference on 3D Vision (3DV)*, pages 32–41. IEEE.
- Nakajima, Y. and Saito, H. (2019). Efficient object-oriented semantic mapping with object detector. *IEEE Access*, 7:3206–3213.
- Rosen, D. M., Mason, J., and Leonard, J. J. (16.05.2016 - 21.05.2016). Towards lifelong feature-based mapping in semi-static environments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1063–1070. IEEE.
- Rusu, R. B. and Cousins, S. (09.05.2011 - 13.05.2011). 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE.
- Solovyev, R., Wang, W., and Gabruseva, T. (2021). Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117.
- Sunderhauf, N., Pham, T. T., Latif, Y., Milford, M., and Reid, I. (24.09.2017 - 28.09.2017). Meaningful maps with object-oriented semantic mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085. IEEE.
- Yi, K. M., Yun, K., Kim, S. W., Chang, H. J., and Choi, J. Y. (23.06.2013 - 28.06.2013). Detection of moving objects with non-stationary cameras in 5.8ms: Bringing motion detection to your mobile device. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 27–34. IEEE.