

Real-Time Multi-Modal Semantic Fusion on Unmanned Aerial Vehicles

Simon Bultmann*, Jan Quenzel*, and Sven Behnke

Abstract—Unmanned aerial vehicles (UAVs) equipped with multiple complementary sensors have tremendous potential for fast autonomous or remote-controlled semantic scene analysis, e.g., for disaster examination.

In this work, we propose a UAV system for real-time semantic inference and fusion of multiple sensor modalities. Semantic segmentation of LiDAR scans and RGB images, as well as object detection on RGB and thermal images, run online onboard the UAV computer using lightweight CNN architectures and embedded inference accelerators. We follow a late fusion approach where semantic information from multiple modalities augments 3D point clouds and image segmentation masks while also generating an allocentric semantic map.

Our system provides augmented semantic images and point clouds with ≈ 9 Hz. We evaluate the integrated system in real-world experiments in an urban environment.

I. INTRODUCTION

Semantic scene understanding is an important prerequisite for solving many tasks with unmanned aerial vehicles (UAVs) or other mobile robots, e.g., for disaster examination in search and rescue scenarios. Modern robotic systems employ a multitude of different sensors to perceive their environment, e.g., 3D LiDAR, RGB(-D) cameras, and thermal cameras, that capture complementary information about the environment. A LiDAR provides accurate range measurements independent of the lighting conditions, while cameras provide dense texture and color in the visible spectrum. Thermal cameras are especially useful in search and rescue missions as they detect persons or other heat sources regardless of lighting or visibility conditions. The combination of all these sensor modalities enables a complete and detailed interpretation of the environment. A semantic map aids inspection tasks [1], perception-aware path planning [2], and increases robustness and accuracy of simultaneous localization and mapping (SLAM) through the exclusion of dynamic objects during scan matching [3].

In this work, we propose a framework for online multi-modal semantic fusion onboard a UAV combining 3D LiDAR range data with 2D RGB and thermal images. An embedded inference accelerator and the integrated GPU (iGPU) run inference online, onboard the UAV for mobile, optimized CNN architectures to obtain pixel- resp. pointwise semantic segmentation for RGB images and LiDAR scans, as well as object bounding box detections on RGB and thermal images.

*: equal contribution.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) in the project “Kompetenzzentrum: Aufbau des Deutschen Rettungsrobotik-Zentrums (A-DRZ)”

Institute for Computer Science VI, Autonomous Intelligent Systems, University of Bonn, Friedrich-Hirzebruch-Allee 8, 53115 Bonn, Germany, {bultmann, quenzel, behnke}@ais.uni-bonn.de

978-1-6654-1213-1/21/\$31.00 ©2021 IEEE

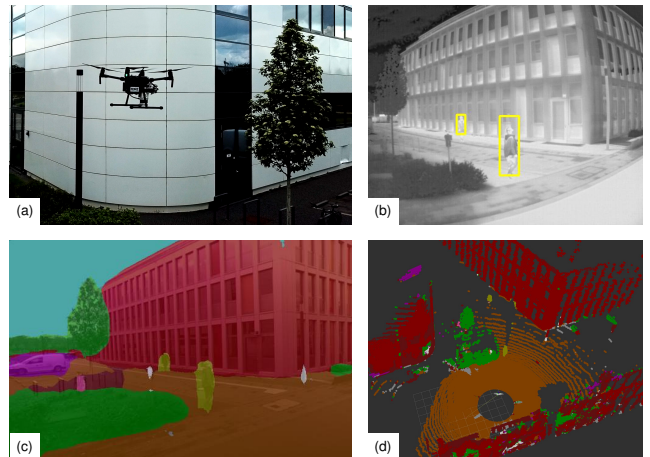


Fig. 1: Semantic perception with UAV (a): (b) Person detections in thermal camera, (c) fused image segmentation and (d) point cloud segmentation.

We aggregate extracted semantics for two different output views: A fused segmentation mask for the RGB image which can, e.g., be streamed to the operator for direct support of their situation awareness, and a semantically labeled point cloud, providing a 3D semantic scene view which is further integrated into an allocentric map. This late fusion approach is beneficial for multi-rate systems, increasing adaptability to changing sensor configurations and enabling pipelining for efficient hardware usage.

In summary, our main contributions are:

- the adaptation of efficient CNN architectures for image and point cloud semantic segmentation and object detection for processing onboard a UAV using embedded inference accelerator and iGPU,
- the fusion of point cloud, RGB, and thermal modalities into a joint image segmentation mask and a semantically labeled 3D point cloud,
- temporal multi-view aggregation of the semantic point cloud and integration into an allocentric map, and
- evaluation of the proposed integrated system with real-world UAV experiments.

II. RELATED WORK

Mobile Lightweight Vision CNNs: Lightweight CNN architectures for computer vision tasks that are efficient and perform well on systems with restricted computational resources, e.g., on mobile or embedded platforms, have become of increasing research interest in recent years. The MobileNet architectures [4], [5] replace classical backbone networks such as ResNets [6] in many vision models while

decreasing the number of parameters and the computational cost significantly, e.g., by replacing standard convolutions with depthwise-separable convolutions—at the expense of a slight reduction in accuracy.

In object detection, single-stage architectures such as SSD [7] or YOLO [8] have proven to be efficient in mobile applications through the use of predefined anchors instead of additional region proposal networks. Zhang et al. [9] further optimize YOLOv3 for usage onboard a UAV. However, the authors evaluate their network, called Slim-YOLOv3, only on a powerful discrete GPU which is not feasible for integration onboard a typical UAV.

Recently, Xiong et al. introduced MobileDets [10] based on the SSD architecture with MobileNet v3 backbone and optimized for embedded inference accelerators such as the Google EdgeTPU [11], which we employ onboard our UAV.

For semantic image segmentation, efficient architectures for inference onboard UAVs have mostly been proposed for specific applications, such as UAV tracking and visual inspection [1] or weed detection for autonomous farming [12]. The DeepLab v3+ architecture [13] shows state-of-the-art performance on large, general datasets while including elements of the MobileNet architectures such as depthwise-separable convolutions for efficient computation. In our work, we employ a DeepLab v3+ model with MobileNet v3 backbone for image segmentation.

For point cloud semantic segmentation, projection-based methods [14], [15], [16] utilize the image-like 2D structure of rotating LiDARs. This allows performing efficient 2D-convolutions and using well-known techniques from image segmentation. The downside of this approach is the restriction to single LiDAR scans in contrast to larger aggregated point clouds [17]. In this work, we adopt the SalsaNext architecture [14], trained on the large-scale SemanticKITTI dataset [18] for autonomous driving, as it shows a good speed-accuracy trade-off.

Semantic Mapping: Many high-level robotic tasks benefit from or require semantic information about the environment. For this, semantic mapping systems build an allocentric semantic environment model, anchored in a fixed, global coordinate frame.

Maturana et al. [19] propose to extend existing digital elevation maps (DEM) with the detection of cars from UAVs.

SemanticFusion [20] models surfaces as surfels where a Gaussian approximates the point distribution. For SLAM, this approach builds on ElasticFusion [21] and requires an RGB-D camera. A CNN generates pixel-wise class probabilities from the color image. Their fusion takes a Bayesian approach assuming that individual segmentations are independent and stores all class probabilities per surfel.

The LiDAR surfel mapping SuMa++ by Chen et al. [3] uses a surfel’s semantic class to further improve the registration accuracy by penalizing inter-class associations during scan matching and surfel update. Here, the projection-based RangeNet++ [15] provides per point class probabilities.

With Recurrent-OctoMap, Sun et al. [22] aim at long-term mapping within changing environments. Here, each cell

within the OctoMap [23] contains an LSTM fusing point-wise semantic features and all LSTMs share weights.

Rosu et al. [24] extract a mesh from an aggregated point cloud. Projection of mesh faces into images enables the transfer from image segmentation to a semantic texture. While projection and fusion happen in real-time, the required mesh generation and UV-unwrapping are done in pre-processing. Since only the argmax class is of interest and to meet GPU memory limitations, the sparse texture retains a small number of classes with high probability and discards all others.

Multi-Modal Semantic Fusion: Mobile robotic systems, such as UAVs or self-driving cars, are often equipped with both camera and LiDAR sensors, as they provide complementary information. A LiDAR accurately measures ranges sparsely and independent of lighting conditions while cameras provide dense textures and colors. Hence, research focused on the fusion of camera and LiDAR for 3D detection and segmentation in the context of autonomous driving.

Xu et al. propose PointFusion [25], a two-stage pipeline for 3D bounding-box detection. It first processes a LiDAR scan with PointNet [26] and an image with ResNet independently, before fusing them on feature level with an MLP.

Meyer et al. [27] take a similar sequential feature-level fusion approach, addressing both 3D object detection and dense segmentation. The feature-level fusion requires representing the LiDAR scan as a range image. Range and color image are cropped to the overlapping field-of-view (FoV), reducing the 360° horizontal FoV of the LiDAR to only 90°.

Vora et al. [28] propose to in-paint point clouds with image segmentation by projecting LiDAR points into the image and assigning segmentation scores of the pixels. A 3D object detection network then processes the augmented point cloud.

In our work, different networks process LiDAR scan, RGB, and thermal images individually. We adopt a projection-based approach similar to [28] for multi-modal fusion in a multi-rate system. When multiple modalities are available, we use a linear combination to merge class probabilities from different sensors. Our mapping integrates augmented point clouds in a sparse voxel hash-map with per voxel full class probabilities. We adapt the Bayesian fusion of SemanticFusion to the logarithmic form for higher precision and greater numerical stability. While being less popular in recent work, such a late fusion approach has important practical advantages for deployment on an integrated robotic system. Different FoVs and data rates are easy to handle and intermediate results, such as image segmentation or detections, are useful as stand-alone outputs. Pipelining also allows reducing latency of sequentially executed individual networks during online operation. Furthermore, the smaller, simpler standard architectures of individual networks are easier to adapt and optimize for the embedded inference accelerators employed in this work.

III. OUR METHOD

A. System Setup

An overview of our UAV system, based on the commercially available DJI Matrice 210 v2 platform, is shown in

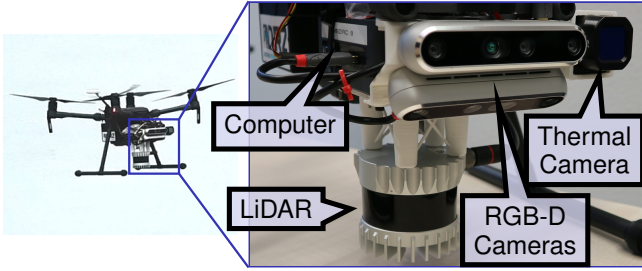


Fig. 2: UAV system setup and hardware design.

Fig. 2. We use an Intel Bean Canyon NUC8i7BEH with Core i7-8559U processor and 32 GB of RAM as the onboard computer. A Google EdgeTPU connects to the NUC over USB 3.0 and accelerates CNN inference together with the Iris Plus Graphics 655 iGPU of the main processor. An Ouster OS0-128 3D-LiDAR with 128 beams, 360° horizontal, and 90° vertical opening angles provides range measurements for 3D perception and odometry. For visual perception, our UAV additionally carries two Intel RealSense D455 RGB-D cameras, mounted on top of each other to increase the vertical field-of-view, and a FLIR ADK thermal camera for, e.g., person detection in search and rescue scenarios.

B. Semantic Perception

An overview of the proposed architecture for multi-modal semantic perception is given in Fig. 3. We detail individual components in the following.

Image Segmentation: We employ the DeepLabv3+ [13] architecture with MobileNet v3 [5] backbone optimized for Google EdgeTPU Accelerator [11] for semantic segmentation. We train the model on the Mapillary Vistas Dataset [29], reducing the labels to the 15 most relevant classes for the envisaged UAV tasks (cf. Fig. 5). We use an input image size of 849×481 px during inference, fitting the 16:9 aspect ratio of our camera.

Object Detection: The recent MobileDet architecture [10] is the basis for our object detection. We train the RGB detector on the COCO dataset [30] for *person*, *vehicle*, and *bicycle* classes with an input resolution of 848×480 px. The thermal object detector uses the same architecture taking one-channel 8-bit gray-scale thermal images at the full camera resolution of 640×512 px as input. We enable automatic gain correction (AGC) for the thermal camera for compatibility with the FLIR ADAS dataset [31]. AGC adapts and scales the 16-bit raw images to 8-bit exploiting the full 8-bit value range. The network is trained on the ADAS dataset, with annotations for *persons*, *vehicles*, and *bicycles*.

Point Cloud Segmentation: We adopt the projection-based SalsaNext architecture [14], pretrained on the large-scale SemanticKITTI [18] dataset that takes advantage of the image-like structure of LiDAR measurements. Subsampling of the OS0 scans by a factor of two in vertical and horizontal directions leads to a network input resolution of 64×512 . The input channels are range, x -, y -, z -coordinate, and intensity, normalized with the mean and standard deviation of the training dataset. Our LiDAR has a significantly larger vertical field-of-view of 90°, compared to the 26.9° vertical

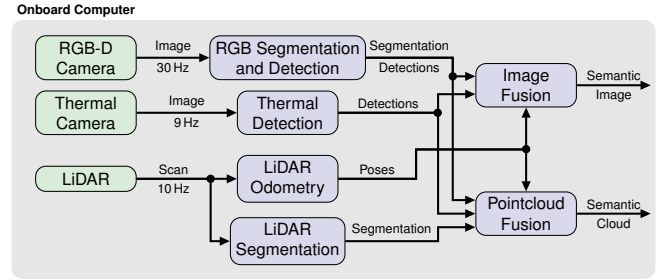


Fig. 3: Perception system overview.

opening angle of the SemanticKITTI dataset. A different laser wavelength also changes the characteristics of intensity and reflections. Hence, we adjust the normalization parameters for z -coordinate and intensity to facilitate the domain adaptation between training and observed data according to the statistics of the test data captured with our sensor setup. The x - and y -coordinate normalization parameters remain the same, as the horizontal field-of-view is identical (360°) for both sensors. Fig. 6 highlights improvements of the segmentation results through the adaptation of the normalization parameters.

Inference Accelerators: We run the CNN model inference on two different accelerators onboard the UAV PC: The Google EdgeTPU [11], attached as an external USB device, and the integrated GPU (iGPU) included in most modern processors which is otherwise unused in our system. The EdgeTPU supports network inference via TensorFlow-lite [32] and requires quantization of the network weights and activations to 8-bit [33]. The iGPU supports inference via the Intel OpenVINO framework [34] in 16- or 32-bit floating-point precision.

C. Multi-Modality Fusion

We adopt a projection-based approach to fuse semantic class scores from image and point cloud. Projection onto the image plane requires the transformation of LiDAR points into the respective camera coordinate frame. As LiDAR and cameras operate with different frame rates, the motion between the respective capture times has to be taken into account. The full transformation chain T from LiDAR to camera frame is:

$$T = {}^{\text{cam}}T_{\text{base}} {}^{\text{base}_{t_c}}T_{\text{base}_{t_l}} {}^{\text{base}}T_{\text{LiDAR}}, \quad (1)$$

using the continuous-time trajectory of the UAV base frame estimated by the LiDAR odometry. Thus, the transformation chain models perspective changes between LiDAR and camera that occur due to dynamic UAV motions.

Bilinear interpolation at the projected point location gives the semantic class scores from image segmentation. A linear combination of the image score and the point score yields the fused class score:

$$c_{\text{fused}} = (1 - w_{\text{img}})c_{\text{LiDAR}} + w_{\text{img}}c_{\text{img}}, \quad (2)$$

with $w_{\text{img}} \in [0, 1]$ and $c \in \mathbb{R}^C$ the vectors of class scores of the CNN output after soft-max of the $C = 15$ classes used in this work (cf. Fig. 5 (e)).

Furthermore, if a projected point falls inside a detection box in either thermal or color images, the detected class is included in the result. We base the detection weight w_{det} on the detector score multiplied with a Gaussian factor with mean at the bounding box center and standard deviation of half the bounding box width resp. height. Again, a linear combination fuses both estimates:

$$\mathbf{c}_{\text{fused_det}} = (1 - w_{\text{det}}) \mathbf{c}_{\text{fused}} + w_{\text{det}} \mathbf{c}_{\text{det}}. \quad (3)$$

Simple projection of all points into the rectangular bounding box will falsely label points in the background as the detected class (cf. Fig. 7). To alleviate this issue, points are clustered w.r.t. their distance in the camera frame before detection fusion. We include only points within the 25% quantile of distances per cluster to focus on foreground objects. The final segmented point cloud includes the full class score vector and the argmax class color per point.

We proceed similarly with the initial image segmentation and detections from RGB and thermal cameras. The RGB-D depth enables projection from RGB to thermal image and temporal smoothing provides a more coherent fused segmentation. For temporal fusion, we project the previous image at time $t - 1$ with its depth into the current frame at time t and perform exponential smoothing:

$$\mathbf{c}_{\text{smoothed_img}_t} = \alpha \circ \mathbf{c}_{\text{img}_t} + (1 - \alpha) \circ \mathbf{c}_{\text{fused_img}_{t-1}}, \quad (4)$$

$$\mathbf{c}_{\text{fused_img}_t} = (1 - w_{\text{det}}) \mathbf{c}_{\text{smoothed_img}_t} + w_{\text{det}} \mathbf{c}_{\text{det}_t}, \quad (5)$$

with the coefficient-wise product \circ .

The smoothing weights α differ between the individual semantic classes. For (potentially) dynamic foreground objects, such as *persons* and *vehicles*, less smoothing is applied (i.e., larger coefficients in α) than for static structures such as buildings and roads. This reduces temporal jitter in the segmentation significantly w.r.t. the initial CNN output and still permits to follow the movement of dynamic objects.

D. Semantic Mapping

MARS LiDAR Odometry [35] provides poses to integrate all augmented point clouds within a common map. A uniform grid subdivides the space into cubic volume elements (voxels). Since a dense voxel grid may require a prohibitively large amount of memory although only sparse access occurs, we use sparse voxel hashing. Each voxel fuses all points in its vicinity probabilistically. Additionally, we compute the mean position. Our fusion scheme follows the reasoning of SemanticFusion [20] to use Bayes' Rule assuming independence between semantic segmentations $P(l_i|X_k)$ for the augmented point cloud X_k with label l_i for class i :

$$P(l_i|X_{1:k}) = \frac{P(l_i|X_{1:k-1}) P(l_i|X_k)}{\sum_i P(l_i|X_{1:k-1}) P(l_i|X_k)}. \quad (6)$$

A naive implementation, as in SemanticFusion, suffers from numerical instability due to finite precision of the multiplication result. In practice, this leads to all class probabilities being close to zero, e.g., when $P(l_i|X_k) \approx 1$, $P(l_j|X_{k+1}) \approx 1$ and $P(l_i|X_{k+1}) \approx 0$, $P(l_j|X_k) \approx 0$ both class-wise products will be almost zero. This results

in a loss of information even after the application of the normalization term and needs continuous reinitialization. Hence, we implement Eq. 6 with log-probabilities:

$$L_{i,1:k-1} = \log(P(l_i|X_{1:k-1})), \quad (7)$$

$$L_{i,k} = \log(P(l_i|X_k)), \quad (8)$$

$$C_{i,1:k} = L_{i,1:k-1} + L_{i,k}, \quad (9)$$

$$M_{1:k} = \max_i (C_{i,1:k}), \quad (10)$$

$$N_{1:k} = \log \left(1 + \sum_j \exp^{C_{j,1:k} - M_{1:k}} \right), \quad (11)$$

$$L_{i,1:k} = C_{i,1:k} - (M_{1:k} + N_{1:k}), \quad (12)$$

$$P(l_i|X_{1:k}) = \exp^{L_{i,1:k}}. \quad (13)$$

Voxels now store $L_{i,1:k}$ instead of $P(l_i|X_{1:k})$. Ideally, we would directly fuse network outputs before soft-max to save additional exp and log evaluations, but since the individual outputs may be arbitrarily scaled, this step is necessary.

An infinite time horizon of the semantic map, fusing all scans, may not be necessary or wanted—depending on the use-case, e.g. for global vs. local planning. Hence, we employ a fixed-size double-ended queue (deque) per voxel for a shorter time horizon of n scans that merges all points per scan. Fusion of per-scan log-probabilities yields the voxels' class probabilities. Older scans are either removed completely or fused into the infinite time horizon estimate.

IV. EVALUATION

We first evaluate inference speed and computational efficiency of the employed CNN models and then show results from our outdoor UAV flights in an urban environment.

A. CNN Model Efficiency

In real-time systems with limited computational resources, such as UAVs, efficiency is of key importance and resources need to be distributed with care between the different system components. Semantic perception, while important for many high-level tasks, has less severe real-time constraints than, e.g., flight control or odometry. It is thus important that the CNN inference uses as few CPU resources as possible to not interfere with the hard real-time constraints of low-level control, localization, and state estimation. For this, we analyze the CPU load of the employed CNNs for object detection and segmentation, depending on the used accelerator. Although the main computational load of inference is distributed to a dedicated accelerator (EdgeTPU or iGPU), the preparation of input data, data transfer, and post-processing require CPU resources. This is handled with differing degrees of efficiency w.r.t. CPU load and depends on the in- and output frame rate, as shown in Fig. 4. Models running on the EdgeTPU produce lower CPU load in all cases while achieving higher or equivalent maximum frame rates. Tab. I shows the average inference latency per model. The LiDAR segmentation is only executed on the iGPU, as the *pixel-shuffle* layer from SalsaNext [14] is not supported

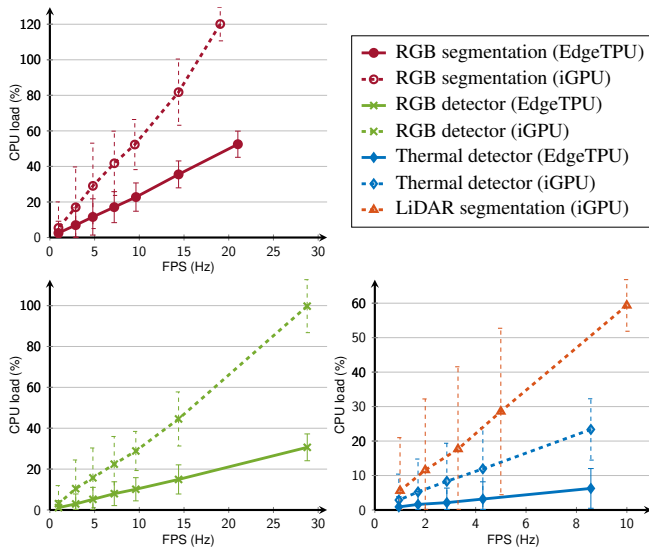


Fig. 4: CPU load of the CNN inference of different models depending on the used accelerator and the output frame rate. The iGPU (dashed lines) results in higher CPU load than the EdgeTPU (solid lines) for all models.

TABLE I: Average inference time on the resp. accelerator.

Model	EdgeTPU	iGPU
RGB Segmentation	40.5 ms	50.0 ms
RGB Detection	17.5 ms	24.0 ms
Thermal Detections	12.0 ms	18.0 ms
LiDAR Segmentation	-	32.0 ms

by the EdgeTPU and the model thus cannot be converted to the required 8-bit quantized format.

For the following experiments, we choose to run the image CNNs on the EdgeTPU, while the LiDAR segmentation runs on the iGPU. Tab. II shows the average computational load and output rate for different combinations of CNNs. As to be expected, the maximum achievable output frame rate drops and CPU load increases with a growing number of vision models used. The computation of RGB segmentation and detection as well as thermal detection achieves an average frame rate of 12.6Hz at a CPU load of about 60%. The inclusion of the image fusion module almost doubles the CPU utilization while the frame rate drops to 9.9Hz. This is due to the transformations and projections necessary to calculate at image resolution for temporal smoothing and to include thermal detection into the fused image segmentation. The total CPU usage for the fusion of both image and point cloud semantics sums up to about 2 CPU cores with an output rate of around 9 Hz.

Reducing the input frequency to the semantic segmentation and detection can free additional resources for other system components if necessary while still providing semantic image and point cloud e.g., at 1 to 5 Hz—sufficient for many high-level tasks like planning or keyframe-based mapping.

B. Outdoor Experiments

In Fig. 5, we show results of semantic image fusion for an exemplary scene from our test flights. Fig. 5 (b) - (d)

TABLE II: Average CPU load and output frame rate of different model combinations. Image segmentation and detection models run on EdgeTPU and point cloud segmentation on iGPU.

point cloud fusion	image fusion	point cloud segm.	thermal det.	RGB det.	RGB segm.	CPU load	avg. FPS (image)	avg. FPS (cloud)
-	-	-	-	-	✓	52.5 %	21.0Hz	-
-	-	-	-	✓	✓	54.2 %	13.2Hz	-
-	-	-	✓	✓	✓	57.3 %	12.6Hz	-
-	✓	-	✓	✓	✓	120.6 %	9.9Hz	-
-	-	✓	✓	✓	✓	116.6 %	12.6Hz	10.0 Hz
-	✓	✓	✓	✓	✓	180.0 %	9.5Hz	10.0 Hz
✓	✓	✓	✓	✓	✓	204.3 %	8.9Hz	9.5 Hz

show the outputs of the individual CNNs. While the large structures are well segmented, the persons are only partially recognized. A bicycle and the person at the right image border are even missed altogether. The RGB detector recognizes all persons and the bicycle. The thermal detector confirms both person detections inside the thermal camera’s field-of-view. The fused output segmentation mask (e) includes all detections together with the initial segmentation mask. All persons and the bicycle are clearly visible.

Fig. 6 shows the point cloud segmentation results for the same scene. Without the adaptation of the normalization parameters, the top half of the buildings are misclassified as vegetation. This is likely due to differing vertical field-of-views of our and the trained LiDAR. In the KITTI dataset [36], the FoV is only $\approx 3^\circ$ upwards and 25° downwards (compared to $\pm 45^\circ$ of our sensor). Furthermore, in SemanticKITTI the top of building structures is rarely visible while the LiDAR often measures treetops at the top end. After normalization adaptation, SalsaNext segments the large structures well within the LiDAR scan (Fig. 6 (b)). Fig. 6 (c) shows the fused point cloud segmentation, combining image segmentation and detections with the initial point cloud segmentation. Persons and small objects are well segmented in the output scan while buildings, vegetation, and the car exhibit less noise in the segmentation when inside the camera FoV. Independent of the normalization, the point cloud network does not detect persons, often misclassifying them as either vegetation or buildings.

In Fig. 7, we show the process of fusing person detections from the RGB or thermal image into the point cloud. The initial segmentation, using only the points projected onto the image segmentation mask, is incomplete and slightly misaligned. The addition of points projected into the bounding box creates many false positives in the background. Our inclusion of only foreground points, after clustering w.r.t. the distance to the sensor (cf. Sec. III-C), results in the entire person being correctly labeled without adding additional mislabeled points in the background.

Fig. 8 depicts the resulting semantic map of the longest test flight with manually annotated semantic labels (a) and with scans either labeled from image segmentation (b) or from fused semantic point clouds (c). The camera-only map

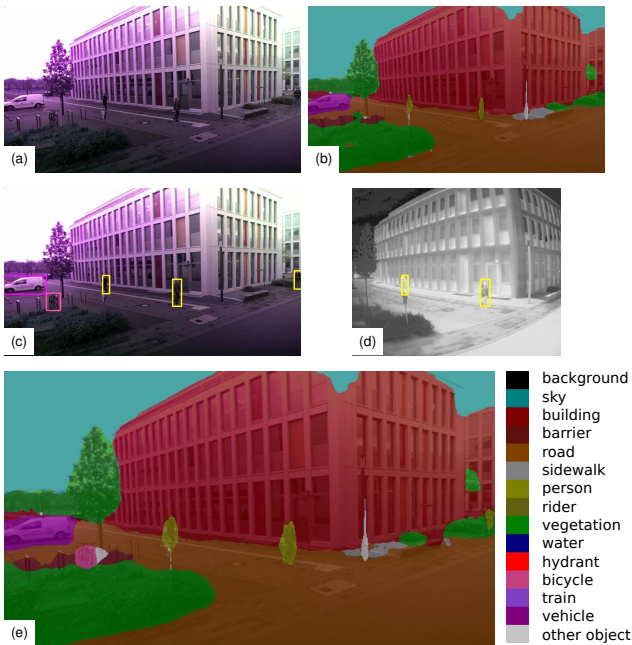


Fig. 5: Semantic interpretation of RGB and thermal images: (a) RGB input image, (b) segmentation and (c) detections. (d) thermal detections. (e) fused segmentation mask. Persons and bicycle, not or only partially segmented in the initial segmentation mask, are fully visible in the fused output.

(b) misses annotations due to the camera’s limited FoV but depicts most classes, such as persons, cars, or vegetation, more accurately since the noisier raw point cloud segmentation is not included. The tracks of moving persons are clearly visible in yellow in both maps. Only the track of the operator, who always stayed behind the UAV and thus was not visible in the camera, is not segmented (b) or mislabeled (c). The direct segmentation of persons or small structures in the LiDAR scans is very noisy due to domain adaptation issues with the employed CNN. The aggregated maps from fused semantic clouds of two further experiments, shown in Fig. 9, underline this issue. In the first test (a) the camera pointed away from people and in the second test (b) more towards the persons within the scene (pointing right w.r.t. the image). Within the camera frustum, person detection works sufficiently well, while they are misclassified as barrier or vegetation elsewhere.

To quantitatively evaluate the coherence of different point cloud segmentations, we calculate the intersection-over-union (IoU):

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c} \quad (14)$$

where TP, FP, and FN are the true positives, false positives, and false negatives, respectively. We compare for each segmented point its argmax class against the corresponding aggregated voxel label and average per class over the whole dataset. Tab. III shows the results for all classes that occur for a significant number of points in our recorded data. We use the manually annotated aggregated semantic map with a voxel size of 25 cm as ground-truth (cf. Fig. 8 (a)). Applying

TABLE III: Average IoU per class (in %) and mean IoU for different point cloud segmentation approaches measured against the manually annotated semantic map.

Segmentation Method	Build.	Road	Veg.	Pers.	Bike	Car	Obj.	Mean
point cloud segmentation w/o adapted normalization	40.4	73.3	3.1	0.2	0.0	2.7	2.2	17.4
point cloud segmentation w. adapted normalization	81.5	73.8	5.9	0.1	0.0	3.6	1.9	23.8
fused semantic cloud	82.9	76.0	20.2	5.8	1.6	6.4	4.9	28.3
fused semantic cloud (reduced to camera FoV)	93.4	66.2	59.9	34.0	16.6	37.6	34.9	48.9
semantic cloud from image overlay (camera FoV)	92.5	76.5	76.7	33.5	17.1	43.5	39.7	54.2

the proposed adaptation of the normalization parameters significantly improves the segmentation of the building class, as the top half of the buildings are correctly labeled (cf. Fig. 6). The fused semantic cloud improves the segmentation coherence for all classes, especially for persons and vegetation. Persons and small objects, such as bicycles, however, are only correctly labeled within the camera FoV. The track of the operator, who always stayed behind the UAV and was not visible in the camera, is misclassified, significantly impacting the mIOU of the person class. Results for the semantic cloud from the projected image segmentation and the fused semantic cloud evaluated for the reduced FoV of the camera show significantly improved mIOU values also for persons, cars, and other foreground objects.

The proposed semantic fusion thus successfully creates a coherently labeled 3D semantic interpretation for the global structure in the full 360° LiDAR FoV and for both global structure and small dynamic objects inside the camera FoV. To improve the accuracy for the difficult semantic classes in the entire FoV, label propagation could further be used for retraining the LiDAR segmentation.

V. CONCLUSION

In this work, we presented a UAV system for semantic image and point cloud analysis as well as multi-modal semantic fusion. The inference of the lightweight CNN models runs onboard the UAV computer, employing an inference accelerator and the integrated GPU of the main processor for computation. The EdgeTPU performs inference in 8-bit quantized mode and showed more efficient CPU usage. The iGPU is more flexible, e.g., to directly run pre-trained models, as it uses 16- or 32-bit floating-point precision and does not require model quantization. We evaluated the system in real-world experiments in an urban environment. The semantic scene analysis provides a 2D image segmentation overlay and a 3D semantically labeled point cloud which is further aggregated into an allocentric semantic map.

The current point cloud segmentation suffers from domain adaptation issues since available large-scale training datasets stem from autonomous driving scenarios with different view-points and sensors more focused towards the ground. Thus, future work includes fine-tuning the laser segmentation with

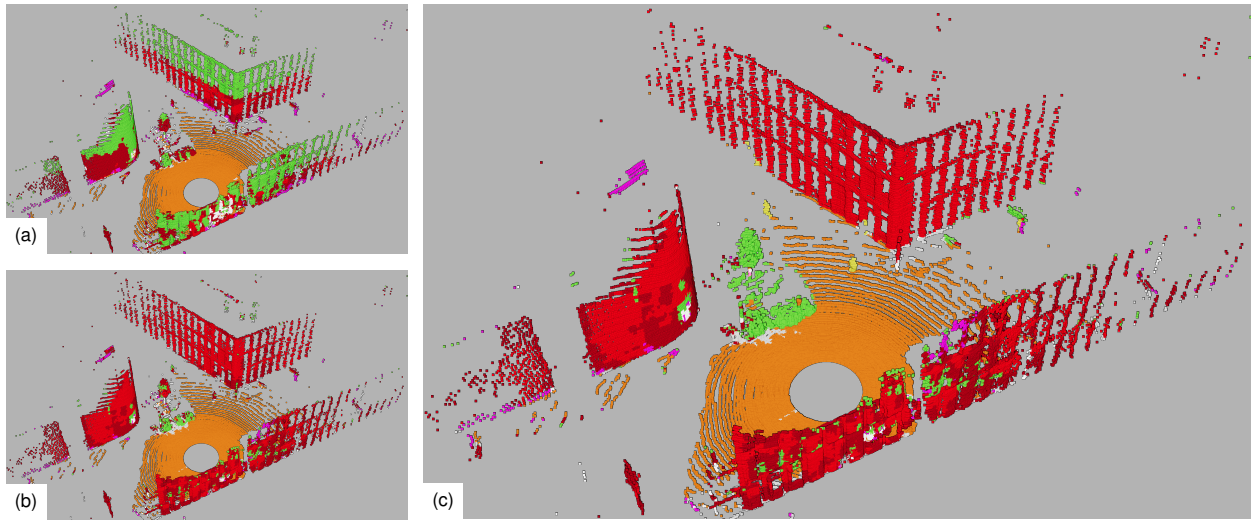


Fig. 6: LiDAR point cloud segmentation: (a) Initial segmentation without adaptation of z and intensity normalization parameters and (b) after adaptation to our dataset mean and std. (c) fused point cloud segmentation. After normalization adaptation, the CNN segments the large structures well within the LiDAR scan but misses small objects. Persons and small structures are well segmented in the fused output scan and buildings, vegetation, and the car exhibit less noise.

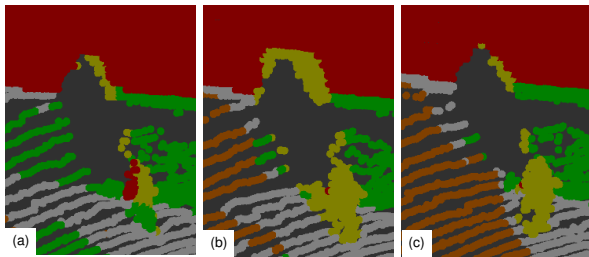


Fig. 7: Person segmentation included into point cloud using (a) image segmentation only, (b) additionally detection bounding boxes, and (c) clustering foreground points within the detections bounding boxes. After clustering, the person is completely segmented without adding additional misclassified points in the background.

simulated data of the employed LiDAR scanner or on data recorded with the UAV and using label propagation to retrain networks with the aggregated semantic map.

REFERENCES

- [1] T. Nguyen, S. S. Shivakumar, I. D. Miller, J. Keller, E. S. Lee, A. Zhou, T. Özarslan, G. Loianno, J. H. Harwood, J. Wozencraft, C. J. Taylor, and V. Kumar, "MAVNet: An effective semantic segmentation micro-network for MAV-based tasks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3908–3915, 2019.
- [2] L. Bartolomei, L. Teixeira, and M. Chli, "Perception-aware path planning for UAVs using semantic segmentation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 5808–5815.
- [3] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based semantic SLAM," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.
- [4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.
- [5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 1314–1324.
- [6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Europ. Conf. on Computer Vision (ECCV)*, 2016, pp. 21–37.
- [8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [9] P. Zhang, Y. Zhong, and X. Li, "SlimYOLOv3: Narrower, faster and better for real-time UAV applications," in *Workshop IEEE Int. Conf. on Computer Vision (ICCV Workshops)*, 2019.
- [10] Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh, and B. Chen, "MobileDet: Searching for object detection architectures for mobile accelerators," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3825–3834.
- [11] Google, "EdgeTPU USB accelerator," <https://coral.ai/docs/accelerator/datasheet>, 2021, accessed: 2021-05-10.
- [12] I. Sa, Z. Chen, M. Popović, R. Khanna, F. Liebisch, J. Nieto, and R. Siegwart, "weedNet: Dense semantic weed classification using multispectral images and MAV for smart farming," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 588–595, 2018.
- [13] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Europ. Conf. on Computer Vision (ECCV)*, 2018.
- [14] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds," in *Int. Symposium on Visual Computing*, 2020, pp. 207–222.
- [15] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet ++: Fast and accurate LiDAR semantic segmentation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 4213–4220.
- [16] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *Europ. Conf. on Computer Vision (ECCV)*, 2020, pp. 1–19.
- [17] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3D object detection from point cloud sequences," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6134–6144.
- [18] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2019.
- [19] D. Maturana, S. Arora, and S. Scherer, "Looking forward: A semantic

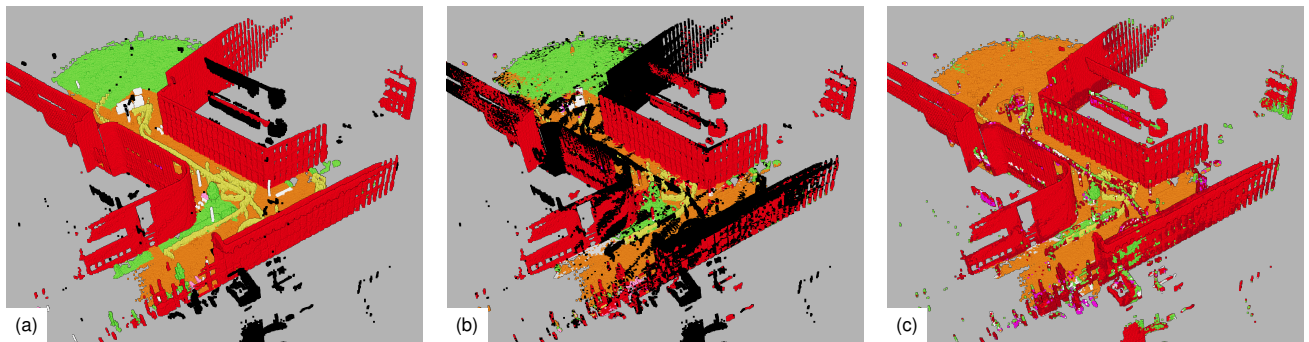


Fig. 8: Semantic map of the longest outdoor flight. (a) manually annotated ground-truth. (b) map created from scans labeled by projected image segmentation only. (c) map created from fused semantic clouds. The camera-only map misses annotations due to the camera’s limited FoV but depicts most classes more accurately, since the noisier raw point cloud segmentation is not included.

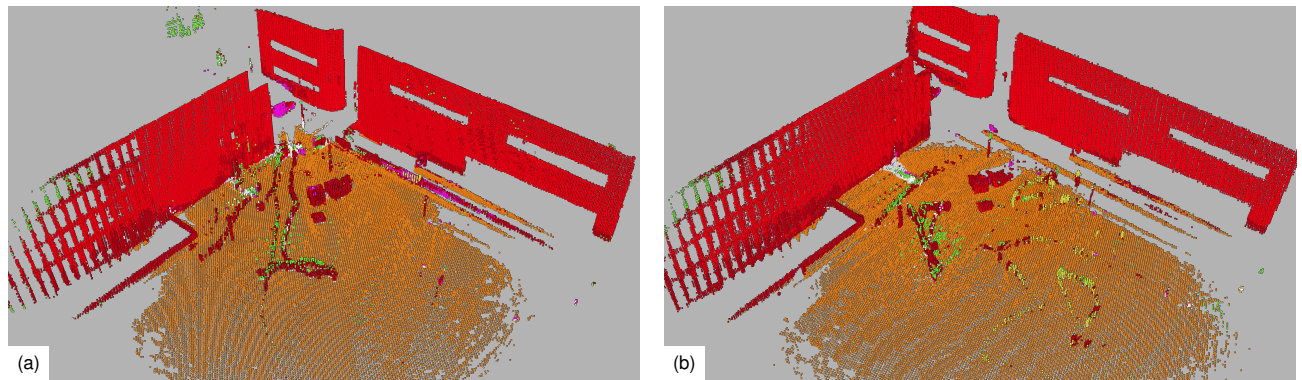


Fig. 9: Aggregated maps from fused semantic clouds of two further experiments with (a) the camera pointed away from and (b) towards people. Within the camera frustum, person detection works sufficiently well, while they are misclassified as barrier or vegetation elsewhere.

mapping system for scouting with micro-aerial vehicles,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 6691–6698.

[20] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “SemanticFusion: Dense 3D semantic mapping with convolutional neural networks,” in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[21] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, “ElasticFusion: Dense SLAM without a pose graph,” in *Robotics: Science and Systems (RSS)*, 2015.

[22] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, “Recurrent-OctoMap: Learning state-based map refinement for long-term semantic mapping with 3D-lidar data,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3749–3756, 2018.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: An efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[24] R. A. Rosu, J. Quenzel, and S. Behnke, “Semi-supervised semantic mapping through label propagation with semantic texture meshes,” *Int. Journal of Computer Vision*, vol. 128, no. 5, pp. 1220–1238, 2020.

[25] D. Xu, D. Anguelov, and A. Jain, “PointFusion: Deep sensor fusion for 3D bounding box estimation,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 244–253.

[26] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, “Sensor fusion for joint 3D object detection and semantic segmentation,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[28] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, “PointPainting: Sequential fusion for 3D object detection,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[29] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kotschieder, “The mapillary vistas dataset for semantic understanding of street scenes,” in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017.

[30] 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 *Europ. Conf. on Computer Vision (ECCV)*, 2014, pp. 740–755.

[31] FLIR, “FLIR thermal dataset for algorithm training,” <https://www.flir.com/oem/adas/adas-dataset-form>, 2021, accessed: 2021-05-10.

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., “TensorFlow: A system for large-scale machine learning,” in *USENIX symposium on operating systems design and implementation (OSDI)*, 2016, pp. 265–283.

[33] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.

[34] Intel, “OpenVINO toolkit overview,” <https://docs.openvino toolkit.org/latest/index.html>, 2021, accessed: 2021-05-10.

[35] J. Quenzel and S. Behnke, “Real-time multi-adaptive-resolution-surfel 6D LiDAR odometry using continuous-time trajectory optimization,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[36] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.