PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

DeepOSM-3D: recognition in aerial LiDAR RGBD imagery

Rajagopal, Abhejit, Stier, Noah, Nelson, William, Chandrasekaran, Shivkumar, Brown, Andrew

Abhejit Rajagopal, Noah Stier, William Nelson, Shivkumar Chandrasekaran, Andrew P. Brown, "DeepOSM-3D: recognition in aerial LiDAR RGBD imagery," Proc. SPIE 11398, Geospatial Informatics X, 113980A (23 April 2020); doi: 10.1117/12.2565585

SPIE.

Event: SPIE Defense + Commercial Sensing, 2020, Online Only, California, United States

DeepOSM-3D: Recognition in Aerial LiDAR RGBD Imagery

Abhejit Rajagopal[a,b,c], Noah Stier[b,c], William Nelson[c], Shivkumar Chandrasekaran[b], and Andrew P. Brown[c]

[a] University of California, San Francisco, USA 94158
[b] University of California, Santa Barbara, USA 93106
[c] Toyon Research Corporation, Goleta, USA 93117

ABSTRACT

In this paper, we present a pipeline and prototype vision system for near-real-time semantic segmentation and classification of objects such as roads, buildings, and vehicles in large high-resolution wide-area real-world aerial LiDAR point-cloud and RGBD imagery. Unlike previous works, which have focused on exploiting ground-based sensors or narrowed the scope to detecting the density of large objects, here we address the full semantic segmentation of aerial LiDAR and RGBD imagery by exploiting crowd-sourced labels that densely canvas each image in the 2015 Dublin dataset.[1] Our results indicate important improvements to detection and segmentation accuracy with the addition of aerial LiDAR over RGB imagery alone, which has important implications for civilian applications such as autonomous navigation and rescue operations. Moreover, the prototype system can segment and search geographic areas as big as 1km^2 in a matter of seconds on commodity hardware with high accuracy ($\geq 90\%$), suggesting the feasibility of real-time scene understanding on small aerial platforms.

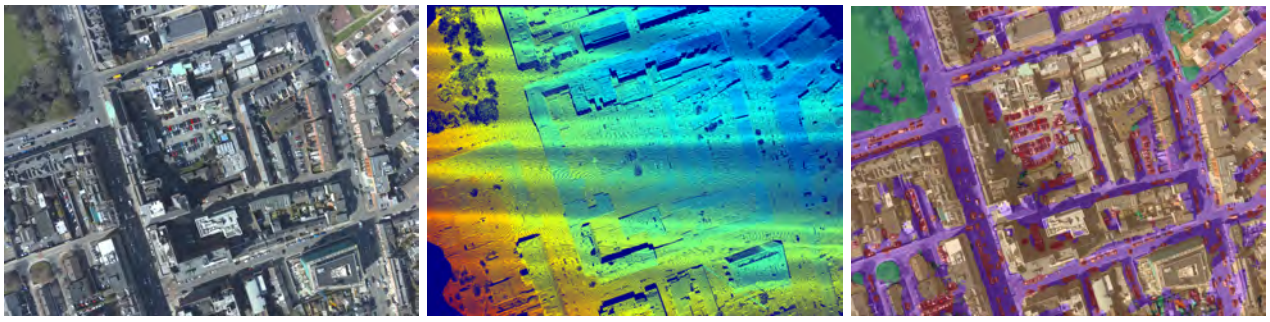

Figure 1. Aerial RGBD imagery (9000px × 6732px) is rendered via automated registration of electro-optic RGB and sparse 3D LiDAR PCD, and fed to a fully-convolutional neural network for semantic and instance segmentation of classes of interest (right).

1. INTRODUCTION

Annotated maps of natural and urban environments are desirable for a number of applications–for example, geographical surveys, urban planning, and search-and-rescue operations. As transistor switching times have decreased, light-detection-and-ranging (LiDAR) has emerged in recent years as a viable option to produce these maps by surveying large landscapes with high accuracy, as opposed to via optical RGB-images alone. LiDAR introduces new information into surveys, primarily a depth channel representing the distance from the LiDAR camera/platform to infrared scatterers in the scene, and this depth information is beneficial for detection and segmentation of objects and terrain. While depth information could be derived from stereographic 3D-reconstruction, LiDAR provides depth measurements at a much higher accuracy and is typically regarded as the gold-standard in structural modeling and mapping applications.

Today, LiDAR systems are very popular in autonomous vehicle platforms, including ground-based and airborne real-time systems. Given the enhanced 3D sensing capabilities of today's manned and unmanned aerial platforms, automated annotation and scene understanding of LiDAR is a compelling computer vision problem. While structural and geometric segmentation codes have proven useful for recognition in small, curated databases of 3D objects,[2,3] these have recently been out-performed by modern deep learning approaches both in terms of

Geospatial Informatics X, edited by Peter J. Doucette, Joshua D. Harguess, Kannappan Palaniappan, Gunasekaran Seetharaman, Proc. of SPIE Vol. 11398, 113980A · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2565585

accuracy and run-time performance.[4–6] However, demonstration of deep neural networks (DNNs) on *wide-area* aerial LiDAR remains in its infancy, primarily due to the absence of a large corpus of labeled training data and efficient algorithms for processing it in a way that enhances overall system performance compared to systems relying solely on more-conventional modalities such as 2D electro-optical (EO) imagery.[7] One reason for the lack of such a dataset is the inherent difficulty of manually recognizing and labeling objects in 3D point clouds.

In this paper we attempt to bridge this gap by outlining a scalable, semi-automated method for labeling wide-area geospatial data such as 3D LiDAR, permitting supervised training of machine learning models. We demonstrate that, where available, public ledgers such as OpenStreetMap (OSM) or Google Maps can be leveraged to generate high-quality semantic segmentation masks for road, building, and natural features (among other interesting classes), that can be used to train and validate models for wide-area detection, localization, and classification. While similar techniques leveraging OSM have been used for building, road, and landcover classification in aerial RGB and SAR imagery,[8–11] extension to aerial LiDAR has been surprisingly limited.[12–14]

In contrast to pure LiDAR point-cloud-based approaches,[6,13,15] our system incorporates pixel-level fusion of LiDAR and RGB similar to prior representations[7,16] and contemporary approaches in ground-based applications.[17,18] This pixel-level fusion is a convenient representation for jointly leveraging geographic and non-geographic label data: in our case, OSM labels and image-plane labels respectively. The joint 2D-3D representation enables 3D data visualization and exploration, while benefiting from the efficiency of 2D image recognition algorithms. The development of this joint representation, however, is a significant engineering challenge, particularly due to the difficulty of registration, since the RGB and LiDAR sensor parameters and telemetry may not be available or accurate, and the data may not have been acquired coincidentally.

Our framework overcomes these challenges to segment and classify `roads`, `vegetation`, `buildings`, and `vehicles` in wide-area RGBD imagery at a rate of 0.3 km^2/s with greater than 90% accuracy using commodity hardware. Further, our results highlight the quantitative benefits of leveraging LiDAR over RGB alone (Fig. 1).

1.1 Related Work

Semantic segmentation and classification of multi-modal geospatial data has been an active research area for more than three decades. Recent developments in neural network-based algorithms have lead to increased interest in exploitation of complimentary modalities, such as the inclusion of SAR reflectance, or LiDAR depth, in addition to RGB EO.[7] provides a comprehensive overview of various data models and the current state-of-the-art applicable to aerial and satellite platforms.

Specifically for LiDAR, Qi *et al*[6] introduced point-set neural networks for semantic segmentation of 3D point-clouds of small 3D objects, indoor scenes, and outdoor ground-based scenes. Recent extensions[15,17,19] have since switched to utilizing 2D projected versions of depth information for greater efficiency and robustness. This latter representation has the benefit of being co-registered with conventional EO imagery, resulting in RGBD representations amenable to real-time navigation applications.[20,21] Unfortunately, in contrast to ground-based applications, there is a dearth in the number of suitable datasets for this task. The 2D and 3D ISPRS Semantic Labeling contests[22] address this to some extent, although data is provided in ortho-ready format and at a very coarse resolution. Thus, the primary bottleneck for exploiting wide-area LiDAR and RGBD imagery lies in the availability of large annotated datasets for training and validation, as opposed to the availability of computer vision methods or systems themselves.[7,18]

To this end, one of the primary goals of this paper is to enable the full exploitation of publicly-available wide-area multi-modal LiDAR datasets such as 2013 Vancouver,[23] 2015 Dublin,[1] 2017 OGRIP,[24] and 2018 District of Colombia,[25] which are all *high resolution* scans of rural and urban terrain with accompanying EO imagery, but with no associated labels for semantic classes of interest. We will use the 2015 Dublin dataset in this paper because the structure of the dataset is typical of many geospatial imagery-collections, and thus these processing algorithms are highly applicable to various other datasets, including other modalities such as EO, synthetic aperture radar (SAR), and ground-based imaging (when camera, orientation, and position information are readily available).

	Area	LiDAR Density	RGBD	Aerial	# Classes
SemanticKITTI	5.2 km^2	878* pt/m^2	✓	✗	25
ISPRS 2D	4.8 km^2	4 pt/m^2	✓	✓	5
Dublin-OSM	4.1 km^2	100-335 pt/m^2	✓	✓	4+

Table 1. Comparison of some relevant LiDAR/RGBD datasets. * composite point-cloud density for KITTI.

1.2 Contributions

The contributions of this paper are as follows:

- We introduce a pipeline for assembling and annotating wide-area 2D RGBD and 3D LiDAR datasets using a variety of labels and coordinate systems, including manual pixel-level annotations and crowd-sourced geographical annotations (e.g. OpenStreetMap).
- We evaluate the performance of deep learning for coarse semantic and instance segmentation of aerial LiDAR and RGBD maps, indicating high-accuracy detections ($\geq 90\%$) in-spite of poor data conformity.
- We provide practical recommendations for applying state-of-the-art deep learning algorithms to the specific sensing context of joint airborne RGB and LiDAR.
- We evaluate the quantitative benefit of aerial LiDAR over RGB imagery alone.

The rest of this paper is organized as follows. In Section 2 we describe our method for automatic labeling of geospatial data such as 3D LiDAR and co-incident 2D electro-optic (EO) imagery. In Section 2.3 we discuss some algorithms and associated learning objectives for this data model. Finally, in Section 3 we discuss the performance of the prototype system, and avenues for future research.

2. DATASET AND MODELING

2.1 Imagery Dataset

In this paper we use the 2015 Dublin LiDAR data collection, which is freely available at https://geo.nyu.edu.[1] The dataset was collected via an airborne platform containing both a full-waveform LiDAR sensor (TopEye system S/N 443), as well as a Phase One imaging system that captured multiple modalities, including geo-referenced RGB imagery, color-infrared imagery, and video data. In this work, we choose to work exclusively with the LiDAR and RGB optical measurements, since these are the most commonly available modalities in geographical surveying and search-and-rescue operations, and have the clearest applicability to modern scenarios, e.g. autonomous navigation.

Data was collected along 41 different flight paths over two sessions at an average altitude of 300m, canvasing a 3km x 3km region around Dublin, Ireland, as shown in Figure 2. EO images were recorded at regular intervals along each flight-path, and LiDAR from each path was integrated, referenced, transformed, and merged into a single LAS point-cloud. Thus, *for each* of the 41 flight paths, the dataset consists of a large number of EO images (50+ images of resolution 9000 x 6732 pixels), and one LAS point-cloud with up to tens of millions of points. In post-processing, these flight-path (FP) point-clouds were also merged into a single "composite" LAS point cloud, that was subsequently tiled by geographic area for processing efficiency. This merged format is the most common in mapping applications, since it provides the densest representation of large stationary objects, such as buildings, roads, and natural features. We experiment with processing the composite format, but we note that for real-time applications, the flight-path LiDAR is more representative in terms of data-sparsity. For comparison, the density of the flight-path LiDAR is roughly 100 points/m^2, whereas the density of the composite

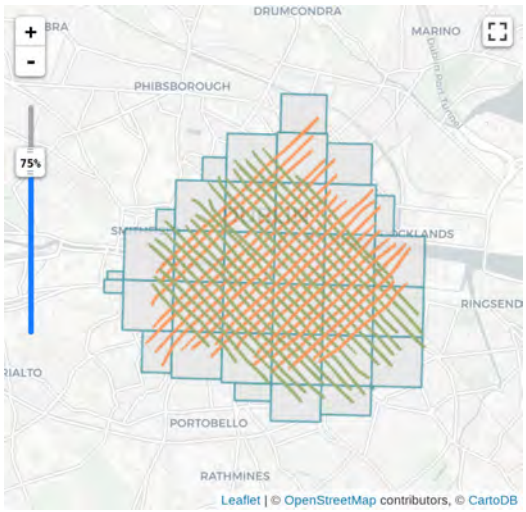

Figure 2. Flight-paths of the data-collection from the 2015 Dublin dataset. Boxes indicate the aforementioned geographic tiles, while the green and orange lines represent different flight-paths.

PCD varies from 100-335 points/m^2 depending on the number of flight paths covering that region. Thus, each flight path image of 60.5M pixels corresponds to upwards of 12M LiDAR points. To make the large images amenable to real-time processing, we downsample the RGB imagery by a factor of 4 prior to registration with the 3D LiDAR PCD.

Once OSM nodes are collected, they are semantically classified (e.g. as `building`, `road`, or `water`) using a combination of OSM-based key-value mappings and keyword mining, and archived as polygon objects for subsequent labeling of images and point-clouds. The simple rules used in this paper to match elements with mutli-class labels are clarified in the Appendix, although more general examples have been demonstrated in prior work.[13] Unfortunately, not all OSM annotations have shape-information; as described in the next section, in some cases we can employ simple heuristics to still leverage portions of these annotations.

2.2 Labeling 3D LiDAR Data and EO Imagery

It is relatively straight-forward to label a georeferenced 3D point-cloud with OSM polygons. When the labels are to be applied to stationary, optically-opaque ground-attached objects (e.g. buildings, structures) it is usually reasonable to assume all points at a geographic index correspond to the same object. Thus, the algorithm should look at each polygon, query the point-cloud for indices that exist within the polygon, and mark each point as belonging to that class. To implement this in the general multi-class setting, for example, one could mark points with a binary vector ($y \in \mathcal{B}^k$) representing assignment to each of k pre-identified classes. In this representation, labeling does not need to include a gridding step (which is computationally expensive, and ultimately prohibitive for wide-area imagery datasets), since clever point-in-polygon algorithms can be employed.[26]

Unfortunately, in OSM, the shape information of roads, walkways, bikepaths, etc. currently does not contain polygon-type information, and instead includes only `LineString` information that represents the start and end of a particular road-segment, along with some auxiliary information on the type of road. In our application, recognition and segmentation of roads is immensely useful, so we choose to infer this geometric information from the OSM tag by heuristically estimating the width of the road (details are provided in the Appendix). However, in estimating these geometries, care must be taken in the labeling algorithm to not replace well-trusted labels (e.g. building annotations) with those that are only estimated. To this end, we opted for non-mutually-exclusive labels that are eventually handled by our loss function and prediction rendering routines (Sec. 2.3).

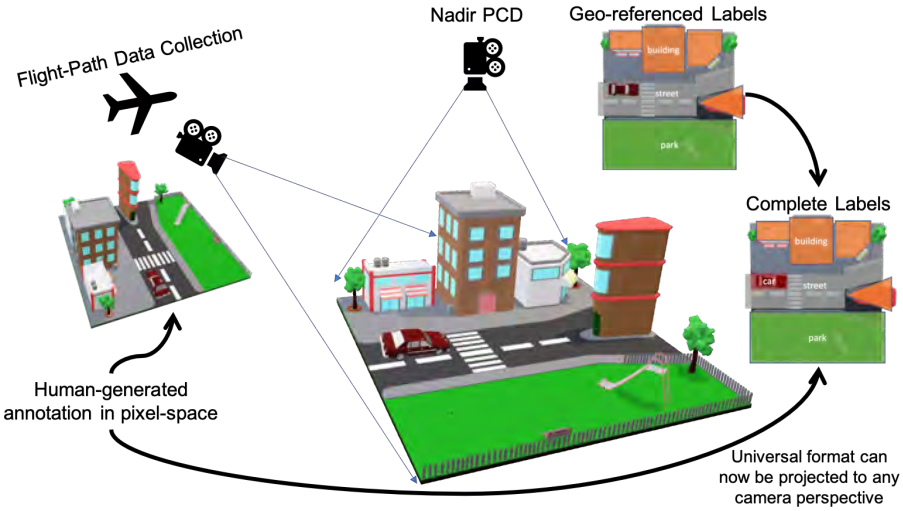

Figure 3. Co-registration between LiDAR PCD and depth images allows sharing of geographic and pixel-space annotations.

For some applications, identification of ephemeral features such as vehicles is also of interest. These would not be stored accurately in a database such as OSM, but can instead be manually identified with relative ease using conventional EO / RGB imagery. To this end, we have manually labeled a subset of the 2015 Dublin data in the native RGB pixel-coordinate system (`row-col`), and transferred these labels into the flight-path PCD representation after applying a semi-automated geo-registration process on the whole dataset.[27,28]

Similarly, using this registration information, we were able to also project earth-referenced PCD information to the perspective of the camera frame, to generate a co-registered depth channel for the EO images, along with corresponding OSM-derived label information (Fig. 3). In this way, we were able to generate a high-quality semantically-labeled LiDAR and RGBD dataset that densely covers the city center of Dublin, Ireland (Fig. 1, 2, 4).

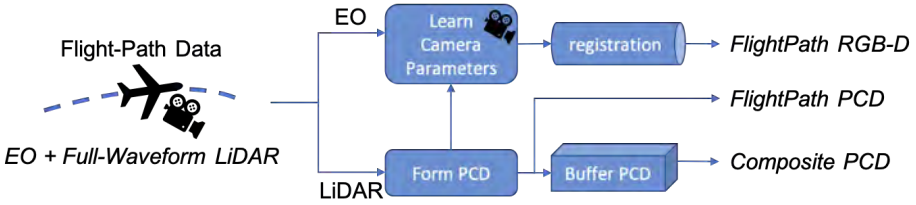

Figure 4. Overview of the data model.

2.3 Data Model

In this paper, we emphasize the construction of the depth-rendered representation of sparse flightpath (as opposed to composite) LiDAR, because it can be done in real time, and it provides a powerful representation that efficiently captures correlations across different spectral bands. In particular, once LiDAR points are projected into the camera frame, RGBD images may be formed and processed using 2D convolutional neural networks, as opposed to voxel-based and point-based networks which can have dramatically higher computational requirements, especially when it is desirable to capture sufficient context within a given a local neighborhood or layer of the network. As discussed earlier, the high LiDAR density of the 2015 Dublin dataset makes depth-renderings much more tractable computationally, and is actually preferable since the spatially-coherent RGBD representation can enhance scene understanding capabilities via a structured representation of color, texture, and shape information.[19,29,30]

2.4 DNN Architecture

As an initial demonstration, we use an adapted version of the U-Net architecture that is popular for segmenting 2D RGB images.[31] Details of our adaptation are provided in the Appendix. In short, we realized an implementation of a fully-convolutional neural network that operates on 2D input images of arbitrary size and depth, and produces output predictions on a grid of equal size to the input. In our implementation for the 2015 Dublin dataset, the size of the receptive field, measured by considering how many input pixels potentially influence an output pixel's value, is at most 26×26 pixels, which corresponds on average to a ground-sample area of 60m^2 in the nadir imagery (Fig. 5). In training, we use tiles of 512×512 pixels, each covering an area of roughly 0.01km^2, but this number is flexible and can be increased for inference (e.g. 2-4x larger).

In our implementation, we noticed that several factors influence the quality of the output segmentation maps. First, the size of the receptive field, which is adjusted by controlling the size and stride of the filter-windows, should be sufficiently large to provide context into the segmentations but also small enough that the network will not learn biased, dataset-specific representations of where objects may lie (e.g. cars are always on labeled roads, next to buildings). Empirically, we found a good range to be $15 \times 15 \leq R \leq 30 \times 30$. Second, the depth of the U-net should be adjusted such that for the smallest input image, the resolution of the latent representation should be enough to be spatially meaningful. In this case, we found that having at least 16x16 pixels in the latent encoding produced desirable results with reasonable training times. Finally, when training this architecture, care must be taken to select examples images at a sufficiently large scale to provide relevance to the output detections (e.g. multiple different classes and boundaries present in the image), while providing sufficient attention to any given object, as to properly penalize any gaps in the output detections.

Note that other neural network architectures, such as DeepLab-v3 and InceptionResnet, were also considered. In our testing, these did not result in sufficient quantitative or perceptible improvements over U-net for this dataset and modality, and are thus not included in this paper. Incorporating an enhanced multi-objective architecture such as Mask-RCNN or similar,[32] on the other hand, is expected to improve localization accuracy beyond the semantic segmentation and grid-based instance detection methods we use in this paper. However, significant changes to the operation and training of these architectures are required to accommodate issues of data and label sparsity, making this a candidate for future work.

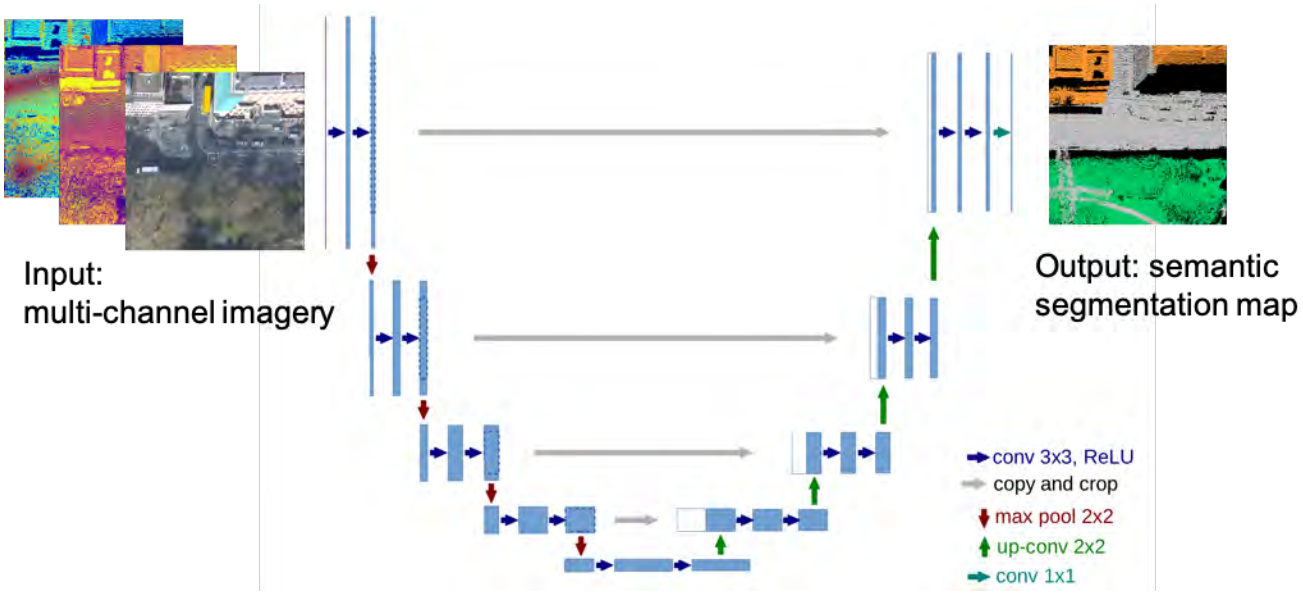

Figure 5. Example operation of a multi-channel extension of the U-Net FCNN architecture[31] for recognition in LiDAR depth- (D) and LiDAR intensity- (I) fused RGBD/I imagery.

2.5 Balanced Loss for Aerial LiDAR Imagery

There are several special considerations for training pattern recognition algorithms in sparse wide-area LiDAR imagery. In our application, the first consideration is that LiDAR returns may not be dense everywhere, resulting in gaps or holes in the depth images, even though the algorithm-predicted semantic labels are typically expected to be geographically dense. This is an obvious problem when LiDAR imagery is unaccompanied by any RGB data, so there is *no basis* to detect objects in a gap region, but it also applies when objects are only partially illuminated by either modality. There are several possibilities to address this issue, such as (a) regressing on dense polygonal labels independent of the point-cloud density, and (b) regressing on point-data alone and ignoring labels outside the labeled pixels/voxels. The issue with these approaches is that they trade specificity (dependence on LiDAR features to indicate a class) for sensitivity (point-wise accuracy), or vice-versa. In other words, regressing on point-data alone encourages improvements in the true positive rate at the expense of the true negative rate, and potentially the false positive rate.

The second consideration is the availability of accurate and dense labels. In this case, shapefile information derived from OSM is generally quite accurate, but may not include annotations for *all* objects in the area (e.g. for `buildings` or `vegetation`).[13] This prevents an accurate measure of the false positive rate for some classes. On the other hand, the OSM-derived `road` labels are spatially overestimated, providing an overly-pessimistic measure of the (pixel-wise) false negative rate. Luckily, manual annotations such as those for `vehicles` are fairly complete and accurate, albeit for a small set of images (typically smaller than the size of available training data). Thus, a good training strategy will use these priors to balance the precision and recall of the multi-class semantic segmentation and instance detection algorithms via an optimal threshold.

To address both of these issues we realized a scheme for training sparse LiDAR depth-rendered RGBD imagery via careful design of the loss function. The loss function is based on the categorical cross-entropy loss that is commonly used to train DNNs and FCNs except with some modality and data-specific modifications, expressed as:

$$-\sum_{i}^{P}\left(z_i \log(\hat{z}_i)\cdot C(z)\cdot \max z_i \cdot \texttt{BOOL}(\min x_i)\right) \tag{1}$$

where $\hat{z}$ and z respectively represent multi-channel prediction and label images of P pixels, $\max z_i$ and $\texttt{BOOL}(\min x_i)$ respectively indicate the presence of a positive label and a LiDAR return at pixel i, and $C(z)$ represents a mask

with the same dimensions as z indicating which classes to consider for a given training image. Note that the non-mutually-exclusive labels are necessary, e.g. for detection of vehicles in parking lots atop buildings.

This loss function first discounts the loss from pixels that lack either a LiDAR point or a positive label. Then, it discounts the negative labels for some classes (namely `vehicles`) in images that are not considered to be labeled to sufficient accuracy. Considering that we are training algorithms for multi-class semantic segmentation of fairly diverse and spatially-broad object classes, this loss function was found to be a sufficient for achieving acceptable false positive and false negative rates (as validated by monitoring these rates for the confidently-labeled `vehicle` class). Note that there is no explicit background class, although there exist unlabeled groundtruth points/pixels.

2.6 Performance Metrics

We evaluate several pixel-wise metrics to quantify algorithm performance. We also introduce the notion of instance detection, because in many remote sensing applications, pixel-level accuracy is not as important as the general detection of an object so it can be flagged for a human operator. In Figures 6 and 7, we illustrate our method of quantifying performance in this regard. For a given image-plane object label, either manually labeled in the image or projected from geographic coordinates, we mark the label as detected if more than half of its pixels are correctly classified. We use this as a heuristic metric for assessing the usefulness of the results.

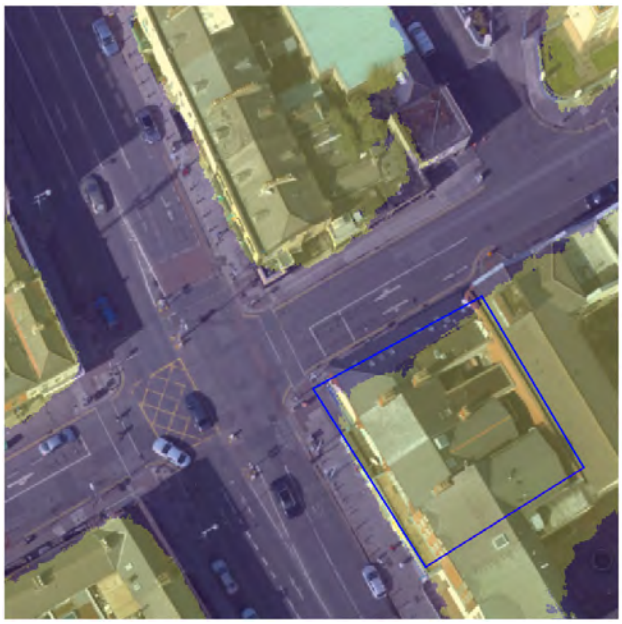

Figure 6. Example of a true positive building instance detection.

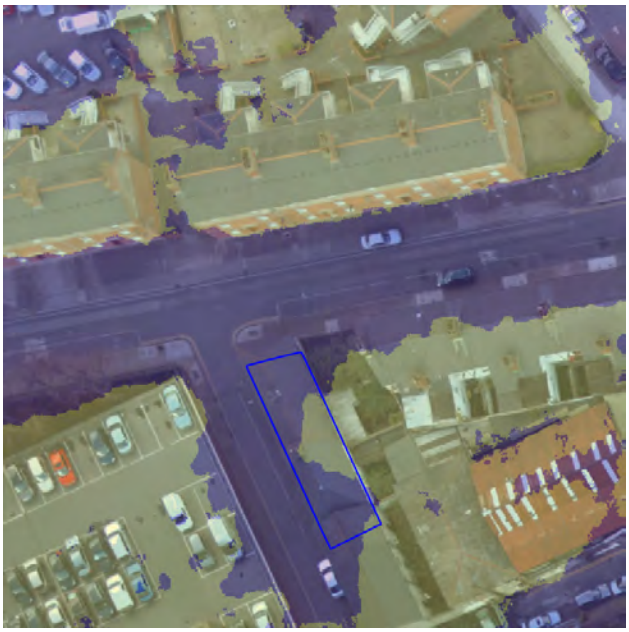

Figure 7. Example of a false negative building instance detection.

We use the following metrics to benchmark the performance of our system, defined here in brief in terms of pixel-wise (PW-) and instance-wise (IW-) true positive (TP), false positive (FP), true negative (TN), and false negative (FN) rates:

- Pixel-wise Accuracy

$$\text{PW-Acc} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- Pixel-wise Sensitivity or True Positive Rate

$$\text{PW-TPR} = \frac{TP}{TP + FN} \tag{3}$$

- Pixel-wise Intersection-Over-Union (IoU)

$$\text{PW-IoU} = \frac{TP}{FP + FN + TP} \tag{4}$$

- Instance Detection Sensitivity or True Positive Rate

$$\text{IW-TPR} = \frac{TP}{TP + FN} \tag{5}$$

- Instance Detection Precision

$$\text{IW-Prec} = \frac{TP}{TP + FP} \tag{6}$$

Note that since this is a non-mutually-exclusive multi-class problem, true positives and true negatives are pixels that are predicted correctly above or below some threshold. Similarly, the instance detection rates are computed based on the a threshold on the percentage of positive detections in a small detection area (empirically chosen to be 30×30 pixels). As explained previously, crowd-sourced OSM-labels *do not* provide a good measure of the IW-FN rate since some annotations may be missing entirely from the ledger for a given region. Finally, we assume mined labels canvas a majority of the dataset, so we discount pixel-wise errors where groundtruth pixels lack a positive label or a projected LiDAR point. We refer readers to the Appendix for more details, and other specifics such as the splitting of the "Train" and "Test" sets used in the performance evaluation.

3. RESULTS AND DISCUSSION

Our pixel-wise and instance detection results using the aforementioned U-net architecture and 2015 Dublin LiDAR dataset are summarized in Table 2. Our results overwhelmingly indicate that the RGBD modality significantly improves the detection of roads, vegetation, buildings, and vehicles in airborne imagery. In particular, detection and segmentation of vehicles is significantly enhanced.

		Road	Vegetation	Building	Vehicle
PW-ACC	D	.788	.925	.842	.834
	RGB	.780	.967	**.896**	.896
	RGBD	**.796**	**.975**	.845	**.932**
PW-TPR	D	.700	.481	.822	.250
	RGB	**.742**	.829	.791	.485
	RGBD	.721	**.900**	**.833**	**.700**
PW-IoU	D	.628	.461	.696	.231
	RGB	.637	.767	.659	.485
	RGBD	**.644**	**.826**	**.703**	**.674**
IW-TPR	D	.688	.360	.832	.290
	RGB	.733	.691	.759	.533
	RGBD	**.869**	**.927**	**.918**	**.918**
IW-Prec	D	-	-	-	.756
	RGB	-	-	-	**1.0**
	RGBD	-	-	-	.980

Table 2. Results: comparison of different performance metrics across input modality and semantic class using U-net-based fully-convolutional neural network on the developed 2015 Dublin-OSM RGBD dataset.

Of course, despite strength in these numbers, in vacuum these metrics are not telling of the full performance of our system since these are not normalized by object volume or occurrence. In general, we visually observe excellent detection accuracy with a very low false positive rate and a strong correspondence with the underlying

RGB imagery, as seen in Figure 8. The semantic segmentation is disassembled in Figures 9 and 10 in order to visualize the multi-class and pixel-wise performance across the various semantic classes. This disassembly is important, since we are not enforcing predictions to be mutually exclusive (e.g. a pixel or ground-point can be simultaneously marked as both `road` and `vehicle`), so it helps in the visual validation of the trained algorithms.

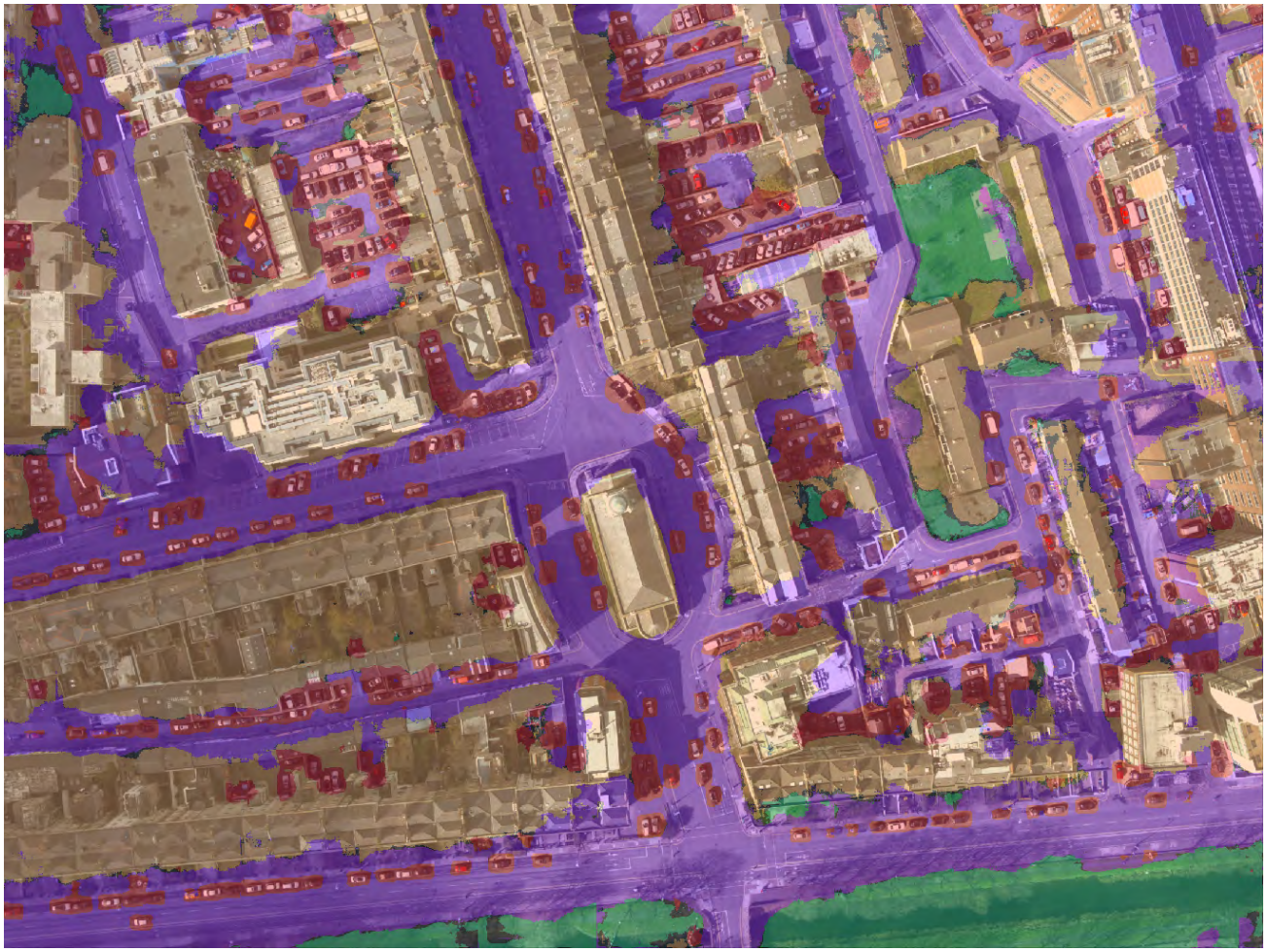

Figure 8. Semantic segmentation prediction results overlaid on native resolution RGB imagery. At each pixel, the color corresponding to the class with the highest activation is visualized. Colors: Purple–roads, Green–vegetation, Orange–buildings, Red–vehicles.

In addition to the IoU segmentation and instance precision scores, the performance of our system can be measured by how well it localizes objects in wide-area imagery at minimal latency. Our simple prototype system is able to process 1km^2 of imagery in under 3 seconds, albeit without exploiting the full 3D morphology of LiDAR PCD. The idea here is that highly performant and relatively inexpensive initial 2D semantic segmentation can queue subsequent fine 3D semantic and instance segmentation algorithms. This provides a massive benefit for processing large, wide-area aerial LiDAR datasets. The proposed camera-rendered representation already serves as a good, fast, and cheap surrogate for a full 3D segmentation, but this can be further improved by specific 3D processing. Ideally, this 3D processing is localized on small regions, or specific objects of interest, as to enhance visualization or discrimination of fine object class. In this vein, future work is focused on jointly refining 2D and 3D instance segmentations, disentangling the dependence on the EO and LiDAR modalities, and validating the robustness to changes in the aspect angle.

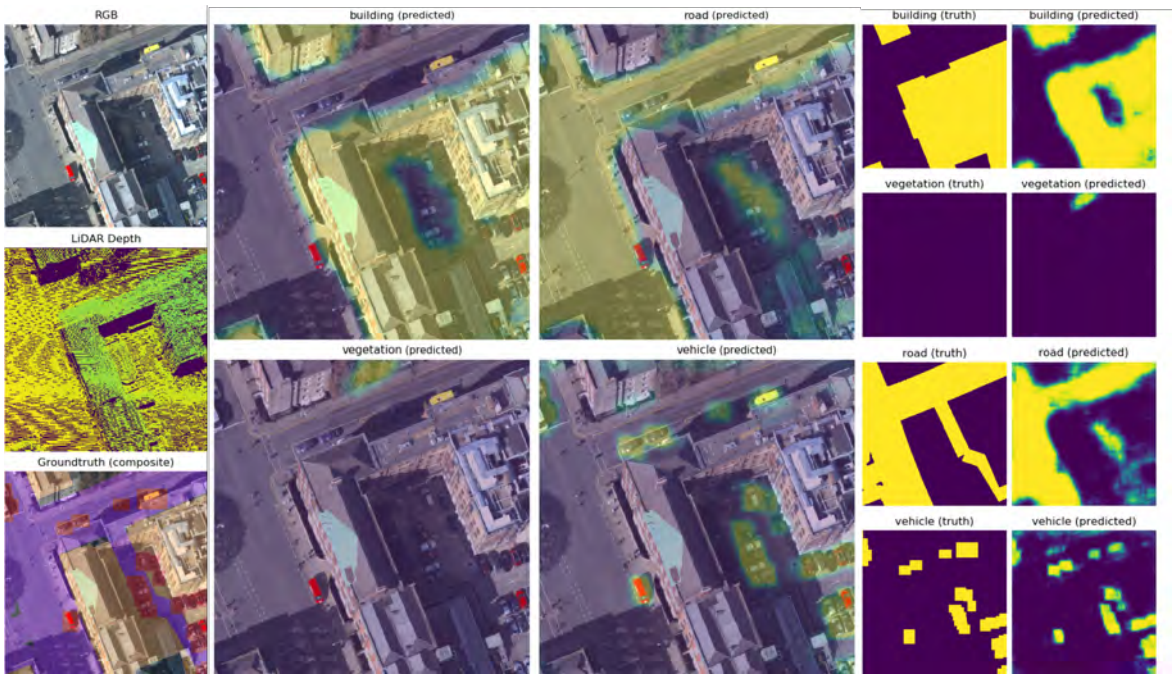

Figure 9. The prototype system exhibits excellent localization performance simultaneously on several classes of interest. Notice that our network correctly identifies trees as `vegetation` in the upper portion of the image and ignores gaps in `buildings`, highlighting enhanced performance over the numerical results reported in Table 2

More generally, the presented techniques suggest interesting extensions to pre-existing datasets, such as those currently used to validate for autonomous-vehicle platforms (e.g. KITTI, comma.ai), since OSM labels can be readily projected into these scenes, augmenting manually-generated segmentation labels. Moreover, with increased interest in image-fusion technologies, generating large wide-area datasets for RGBD imagery is crucial to developing robust and generalizable architectures for real-time scene understanding on these platforms.[18,20,33] In this spirit, the presented technique and results on an airborne platform can be easily applied to ground-based platforms when alignment between EO and (LiDAR) depth can be achieved.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Roman Ilin and Dr. Benjamin D. Robinson for their helpful suggestions, insight, and support of our work under Air Force contracts FA8650-16-M-1784 and FA8650-18-C-1137.

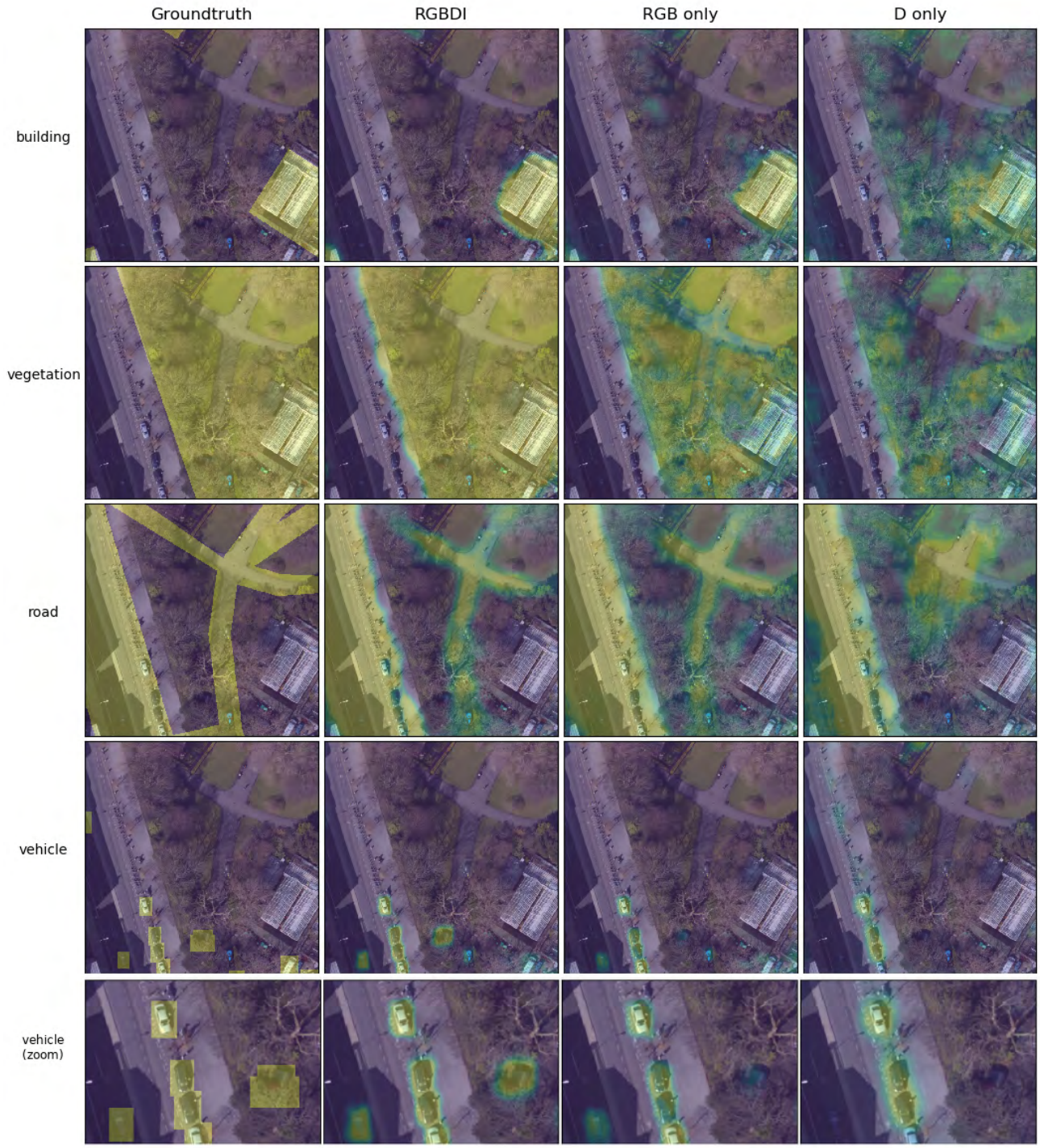

Figure 10. A comparison of semantic segmentation across modalities for an occluded area. Notice that the RGBD-network is able to detect a extra car through a tree, whereas the RGB and Depth-only networks could not find sufficient evidence to make a declaration there. The last row is a crop and zoom of the vehicle channel to emphasize the comparison.

REFERENCES

[1] Laefer, D. F., Abuwarda, S., Vo, A.-V., Truong-Hong, L., and Gharibi, H., "2015 aerial laser and photogrammetry survey of dublin city collection record," *New York University, Tech. Rep* (2017).

[2] Zermas, D., Izzat, I., and Papanikolopoulos, N., "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in [*Robotics and Automation (ICRA), 2017 IEEE International Conference on*], 5067–5073, IEEE (2017).

[3] Kusenbach, M., Himmelsbach, M., and Wuensche, H.-J., "A new geometric 3d lidar feature for model creation and classification of moving objects," in [*Intelligent Vehicles Symposium (IV), 2016 IEEE*], 272–278, IEEE (2016).

[4] Wu, B., Wan, A., Yue, X., and Keutzer, K., "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in [*2018 IEEE International Conference on Robotics and Automation (ICRA)*], 1887–1893, IEEE (2018).

[5] Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., and Posner, I., "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," in [*Robotics and Automation (ICRA), 2017 IEEE International Conference on*], 1355–1361, IEEE (2017).

[6] Qi, C. R., Yi, L., Su, H., and Guibas, L. J., "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in [*Advances in Neural Information Processing Systems*], 5099–5108 (2017).

[7] Ghamisi, P., Rasti, B., Yokoya, N., Wang, Q., Hofle, B., Bruzzone, L., Bovolo, F., Chi, M., Anders, K., Gloaguen, R., et al., "Multisource and multitemporal data fusion in remote sensing: A comprehensive review of the state of the art," *IEEE Geoscience and Remote Sensing Magazine* **7**(1), 6–39 (2019).

[8] Johnson, A., "Deeposm," (2016).

[9] Yuan, J., "Learning building extraction in aerial scenes with convolutional networks," *IEEE transactions on pattern analysis and machine intelligence* **40**(11), 2793–2798 (2017).

[10] Audebert, N., Le Saux, B., and Lefèvre, S., "Joint learning from earth observation and openstreetmap data to get faster better semantic maps," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*], 67–75 (2017).

[11] Shahzad, M., Maurer, M., Fraundorfer, F., Wang, Y., and Zhu, X. X., "Buildings detection in vhr sar images using fully convolution neural networks," *arXiv preprint arXiv:1808.06155* (2018).

[12] Cabezas, R., Straub, J., and Fisher, J. W., "Semantically-aware aerial reconstruction from multi-modal data," in [*Proceedings of the IEEE International Conference on Computer Vision*], 2156–2164 (2015).

[13] Rajagopal, A., Chellappan, K., Chandrasekaran, S., and Brown, A. P., "A machine learning pipeline for automated registration and classification of 3d lidar data," in [*Geospatial Informatics, Fusion, and Motion Video Analytics VII*], **10199**, 101990D, International Society for Optics and Photonics (2017).

[14] Audebert, N., Le Saux, B., and Lefèvre, S., "Beyond rgb: Very high resolution urban remote sensing with multimodal deep networks," *ISPRS Journal of Photogrammetry and Remote Sensing* **140**, 20–32 (2018).

[15] Zhou, Y. and Tuzel, O., "Voxelnet: End-to-end learning for point cloud based 3d object detection," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 4490–4499 (2018).

[16] Schulter, S., Zhai, M., Jacobs, N., and Chandraker, M., "Learning to look around objects for top-view representations of outdoor scenes," in [*Proceedings of the European Conference on Computer Vision (ECCV)*], 787–802 (2018).

[17] Milioto, A., Vizzo, I., Behley, J., and Stachniss, C., "Rangenet++: Fast and accurate lidar semantic segmentation," in [*Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*], (2019).

[18] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J., "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in [*Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*], (2019).

[19] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J., "Frustum pointnets for 3d object detection from rgb-d data," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 918–927 (2018).

[20] Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N., "Visual odometry and mapping for autonomous flight using an rgb-d camera," in [*Robotics Research*], 235–252, Springer (2017).

[21] Amado, J. A. D., Gomes, I. P., Amaro, J., Wolf, D. F., and Osório, F. S., "End-to-end deep learning applied in autonomous navigation using multi-cameras system with rgb and depth images," in [*2019 IEEE Intelligent Vehicles Symposium (IV)*], 1626–1631, IEEE (2019).

[22] Niemeyer, J., Rottensteiner, F., and Soergel, U., "Contextual classification of lidar data and building object detection in urban areas," *ISPRS journal of photogrammetry and remote sensing* **87**, 152–165 (2014).

[23] City of Vancouver, "2013 dublin lidar dataset," (2013).

[24] Ohio Geographicall Referenced Information Program, "Osip lidar," (2017).

[25] Office of the Chief Technology Officer (OCTO), District of Columbia GIS program, "Lidar - dc point cloud - 2018," (2018).

[26] Murta, A., "Gpc: General polygon clipper library," *Astrophysics Source Code Library* (2015).

[27] Ding, M., Lyngbaek, K., and Zakhor, A., "Automatic registration of aerial imagery with untextured 3d lidar models," in [*Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*], 1–8, IEEE (2008).

[28] Brown, A. P., Sheffler, M. J., and Dunn, K. E., "Persistent electro-optical/infrared wide-area sensor exploitation," in [*Evolutionary and Bio-Inspired Computation: Theory and Applications VI*], **8402**, 840206, International Society for Optics and Photonics (2012).

[29] Meyer, G. P., Charland, J., Hegde, D., Laddha, A., and Vallespi-Gonzalez, C., "Sensor fusion for joint 3d object detection and semantic segmentation," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*], (2019).

[30] Caltagirone, L., Bellone, M., Svensson, L., and Wahde, M., "Lidar–camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems* **111**, 125–131 (2019).

[31] Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," in [*International Conference on Medical image computing and computer-assisted intervention*], 234–241, Springer (2015).

[32] Novotny, D., Albanie, S., Larlus, D., and Vedaldi, A., "Semi-convolutional operators for instance segmentation," in [*Proceedings of the European Conference on Computer Vision (ECCV)*], 86–102 (2018).

[33] Menze, M. and Geiger, A., "Object scene flow for autonomous vehicles," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 3061–3070 (2015).

Appendix

3.1 Data Labeling

3.1.1 OSM Data

The OSM data was downloaded from a Geofabrik server. Nightly-OSM data is available freely for download from: https://download.geofabrik.de/europe/ireland-and-northern-ireland.html. Processing the ESRI shape-file (SHP) format is often the most convenient, and the can be achieved using open-source GDAL software, e.g. via Python. While this represents a curated list (generated in a semi-automated fashion), there are several other OpenStreetMap databases (e.g. OverpassTurbo: https://overpass-turbo.eu/) which can be used to equivalently collect data.

3.1.2 Tag Mining and Classification

All the leaf nodes can be inspected to derive semantic classifications for corresponding geographic areas. For the 2015 Dublin dataset there is a fairly intuitive mapping based on the OSM classes, with some exceptions:

1. Universities were not labeled as `buildings` because their OSM-contours are often not confined to building edges.
2. Shops were labeled as `buildings`.
3. Highways in covered tunnels were not labeled as `roads` because they are not visible to either the EO or LiDAR modality.

Note that although there are numerous `water` features in the Dublin dataset (e.g. waterways, rivers), this was not used in our experiments due to the lack of a representative number of examples.

3.1.3 Geometry Inference from OSM Tags

Road widths were estimated for Dublin, Ireland as follows:

OSM Class	Sub-type	Width (meters)
Highway	Primary	15
Highway	Secondary	15
Highway	Tertiary	10
Highway	–	7

Table 3. Estimated road widths.

3.1.4 Rendering

Wide-area results shown in the paper are rendered using a simple `argmax` operator on the label dimension, when at least one class is above a specified threshold. This enforces a label-hierarchy that is defined by the order in the classification vector (0–green–*Park*, 1–orange–*Building*, 2–purple–*Road*, 3–red–*Vehicle*). This is not a perfect system, but it provides many intuitive designations; for example, `vehicles` appear above `roads` that may occur atop or between `buildings`. Note that unlabeled pixels are rendered in black, but we found 100% of the predicted 2D image was labeled by the network (i.e. no holes).

3.1.5 Train-Test Split

Of the 41 available flight paths, we selected 31 for use in this paper. Of these 31, we manually annotated 9 with `vehicle` polygons using a custom in-house software labeling tool. Seven of these flight paths were used for training and the remaining used for testing. A majority of these polygons are bounding boxes, while a smaller number are free polygon shapes.

The training and testing data were split geographically over the extent of the dataset, such that ground-points (latitude, longitude) that are in a geographic bounding box corresponding to the optical (EO) or LiDAR field of view during a flight path appear mutually-exclusively in either the train or test set. Of the available EO images, roughly 80% of them were selected for training (with corresponding LiDAR points) and the remaining were used for testing.

3.2 Network Details

3.2.1 U-Net Architecture

We used a standard implementation in Keras with a Tensorflow backend. We will publish the precise number of layers and filters that we used after the review period (smaller than the original U-net), but these were standard choices made to fit the network on commodity GPUs for training.

3.2.2 Loss functional

In this paper we employ cross-entropy loss. We use a validity mask to remove the loss contribution from pixels where there is no input LiDAR data or no positive groundtruth. This prevents the network from over-fitting at pixels where there was no information by which to make a successful update. Specifically, the loss was computed as:

$$J = -\frac{\sum_{nij} \mathcal{I}_{nij} \sum_{c=1}^{C} \mathbb{1}\left(y_{nij} = c\right) \log p_{nij,c}}{\sum_{nij} \mathcal{I}_{nij}} \tag{7}$$

where C is the total number of classes, y_{nij} is the the groud truth class label of the pixel in the ith row and jth column of the nth image, $p_{nij,c}$ is the predicted probability that the similarly labeled pixel belongs to class c, and $\mathcal{I}_{nij}$ is a validity mask which is 1 if input LiDAR data *and* groundtruth label was gathered at the specified pixel and 0 otherwise.

Training was performed via the standard Adam's optimizer in Keras/Tensorflow with a learning rate of 0.001 and a batch size of 20.

3.3 Performance Details

3.3.1 Pixel-wise Detection Rates

The pixel-wise detection rates were computed using a threshold of $\tau_1 = 0.2$ on the network output for each class. This number was determined by benchmarking the true and false positive rate for the various classes considered. Pixels were only counted if at least one positive label was present, as to produce overly pessimistic numbers for labels missing from the OSM-ledger. This was balanced with the detection rates for `vehicles`, for which we have a more confidence in the true/false designation. Note that, even for vehicles, the polygon is often overestimated (e.g. when the groundtruth is a bounding box).

3.3.2 Instance-wise Detection Rates

The instance-wise detection rates were computed using a fill-threshold of $\tau_2 = 0.2$ for all classes. That is, for every 30×30 pixel sliding window, the ratio of number of detected pixels of a given class (e.g. using the pixel-wise threshold τ_1) to the total number of pixels in the window, was computed and used to determine whether the window contained an instance. The number τ_2 was picked based on measuring the instnace-wise true positive rate as a function of the fill-threshold for all the classes of interest, and balancing this figure with the instance-wise false positive rate for a class with all known instances (`vehicles`). The size of the window was selected based on the median size of the smallest feature (`vehicles`).

3.3.3 Hardware

Training was performed on a computer with an Intel 8th-generation i9 CPU, 96GB RAM, and a nVidia Titan RTX (24GB) GPU. Performance testing was performed on a computer with an 7th-generation Intel i7 CPU, 32GB RAM, and a nVidia GTX 1080 (8GB) GPU.

3.3.4 Results

Additional visualizations of our results are provided in Figures 11-13.

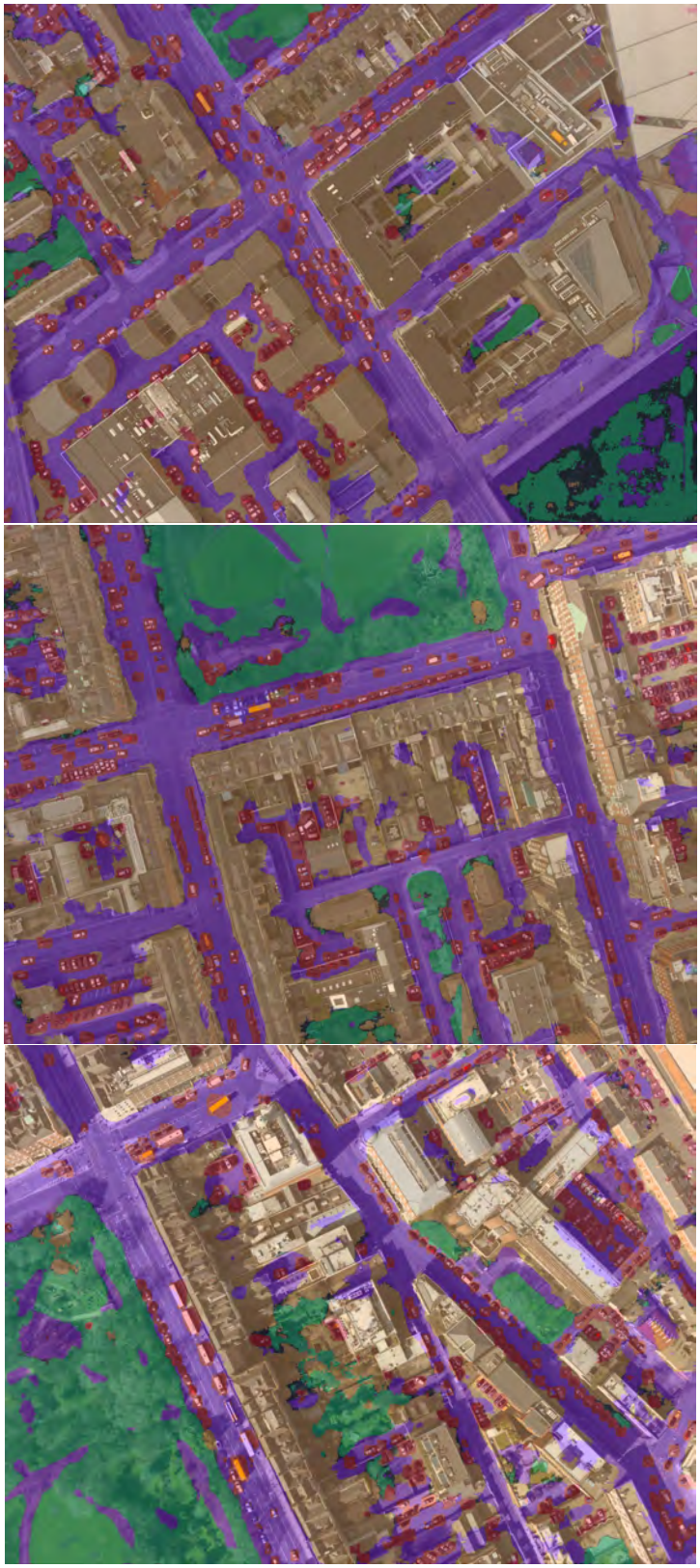

Figure 11. Wide-aperture visualization of semantic segmentation output.

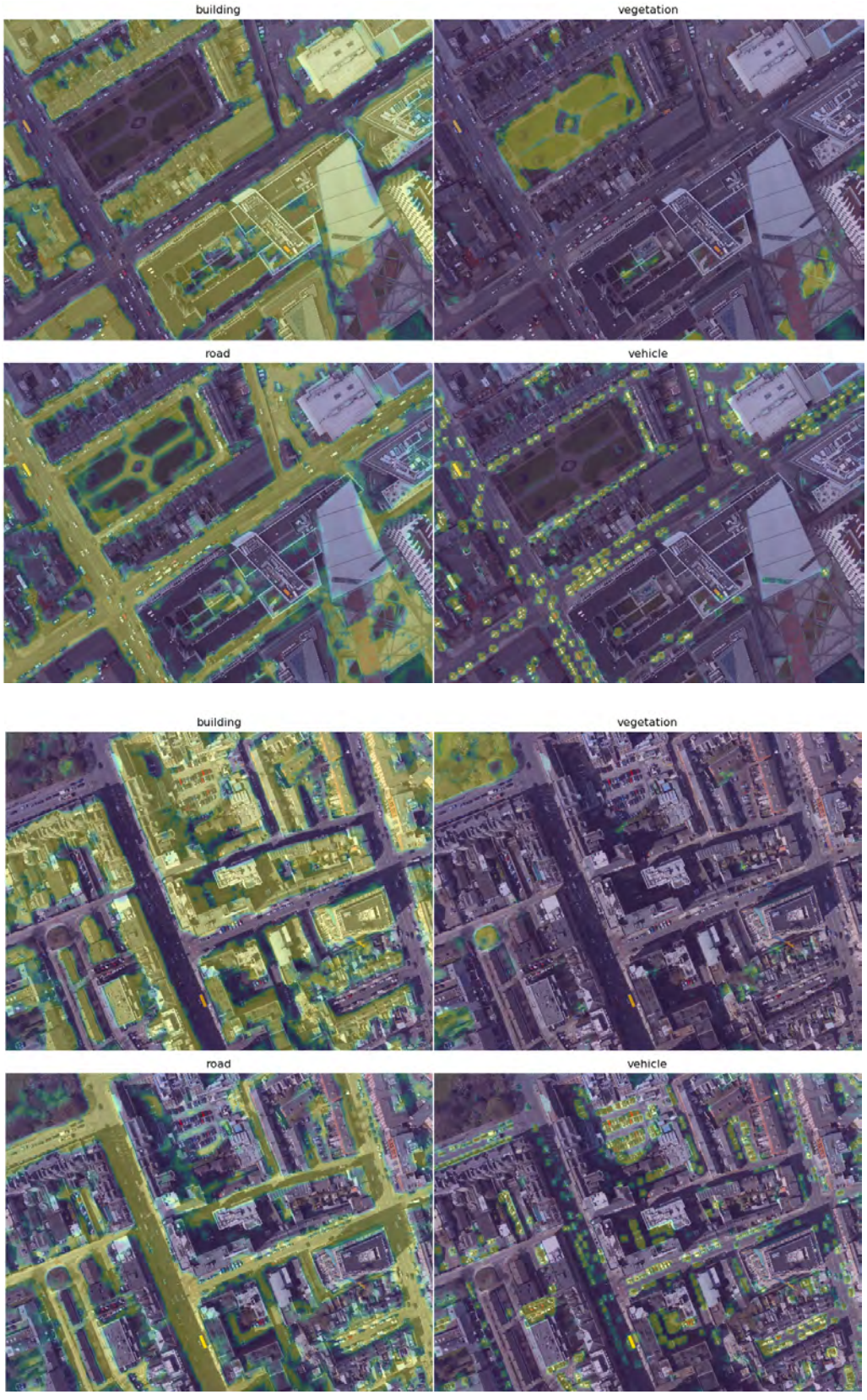

Figure 12. Breakdown of two wide-area multi-class semantic segmentation examples.

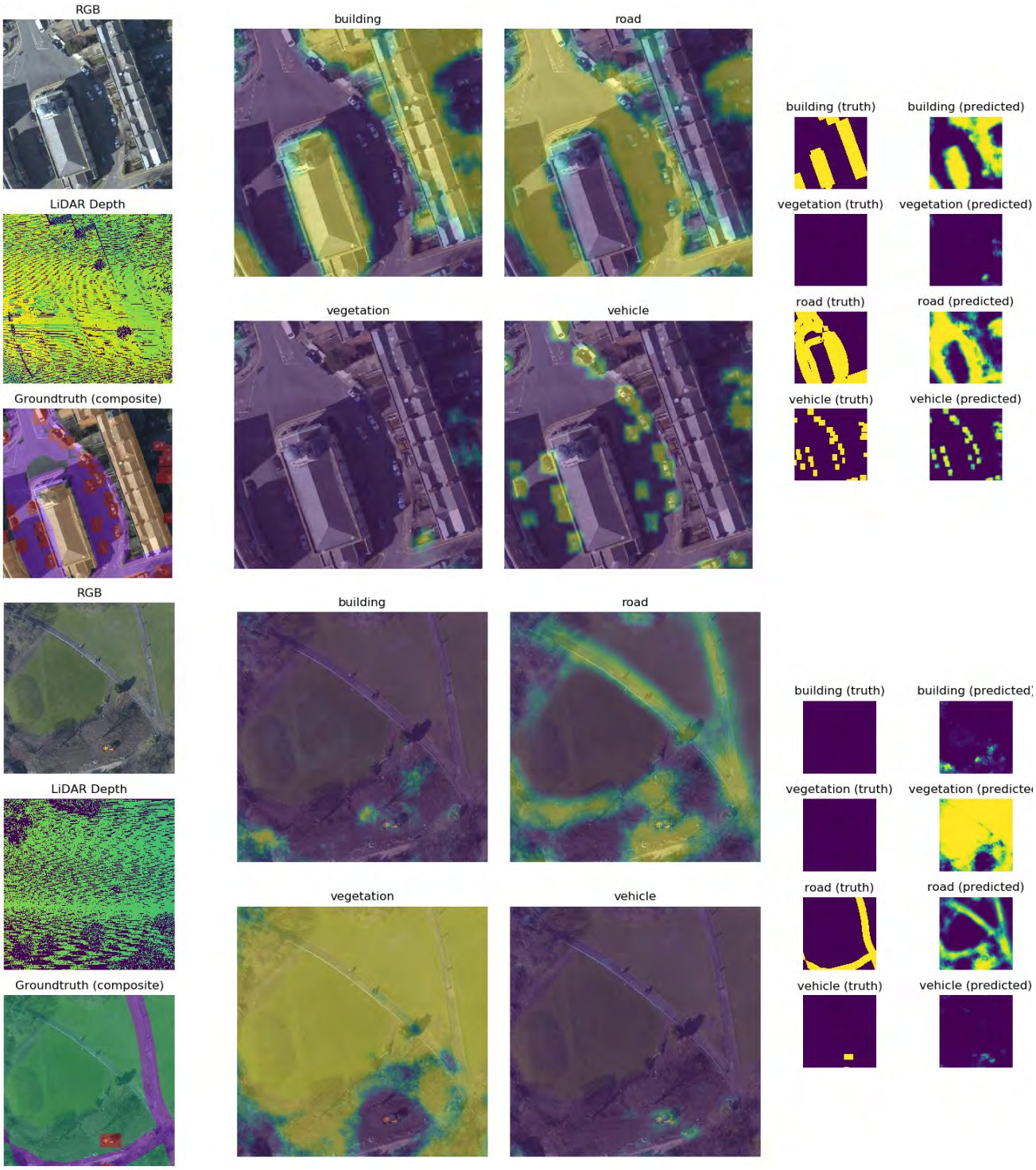

Figure 13. Close-up breakdown of two multi-class semantic segmentation examples.