Looking Forward: A Semantic Mapping System for Scouting with Micro-Aerial Vehicles

Daniel Maturana[1], Sankalp Arora[1] and Sebastian Scherer[1]

***Abstract*— The last decade has seen a massive growth in applications for Micro-Aerial Vehicles (MAVs), due in large part to their versatility for data gathering with cameras, LiDAR and various other sensors. Their ability to quickly go from assessing large spaces from a high vantage points to flying in close to capture high-resolution data makes them invaluable for applications where we are interested in a specific target with an *a priori* unknown location, e.g. survivors in disaster response scenarios, vehicles in surveillance, animals in wildlife monitoring, etc., a task we will refer to *scouting*. Our ultimate goal is to enable MAVs to perform autonomous scouting. In this paper, we describe a *semantic mapping* system designed to support this goal. The system maintains a 2.5D map describing its belief about the location of semantic classes of interest, using forward-looking cameras and state estimation. The map is continuously updated on the fly, using only onboard processing. The system couples a deep learning 2D semantic segmentation algorithm with a novel mapping method to project and aggregate the 2D semantic measurements into a global 2.5D grid map. We train and evaluate our segmentation method on a novel dataset of cars labelled in oblique aerial imagery. We also study the performance of the mapping system in isolation. Finally, we show the integrated system performing a fully autonomous car scouting mission in the field.**

I. Introduction

Micro-Aerial Vehicles (MAVs) can quickly and inexpensively gather information with cameras, LiDAR and various other sensors, due to their agility. This makes them invaluable for applications such as search and rescue, infrastructure inspection, surveillance, crop and wildlife monitoring, etc.

A common trend in these applications is that not all possible locations are of equal value; we are usually more interested in gathering information about specific targets, such as survivors, vehicles, animals, etc. Often, we do not know in advance the location of these targets, making it necessary to locate them before more detailed inspection. For example, in a disaster scenario, we might be interested in searching for survivors and then approach them to capture high-resolution images. Equipped with cameras, UAVs are able to switch from viewing large spaces at a distance to flying in closely to obtain more accurate information. This capability of gaining information at different scales makes UAVs excellent for the aforementioned applications. We will refer to the overall task of searching and gathering data for a semantic class of interest as scouting (fig. 1). Our goal is to create a system to enable MAVs to perform effective general-purpose autonomous scouting.

[1]The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA `dimatura, asankalp, basti@cmu.edu`

Towards this goal, we study the more concrete scenario depicted in fig. 7. In this scenario, we wish to find any cars within a predefined region and capture high-resolution imagery (e.g., for 3D reconstruction). The location, number, and appearance of cars, if any, is not known *a priori*. We have a limited power budget, equivalent to around 10 minutes of flight time.

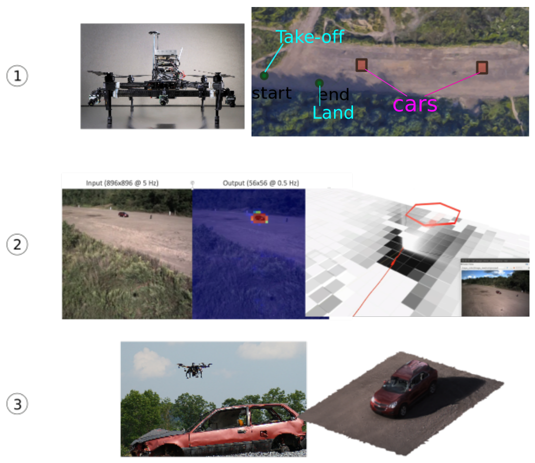


Fig. 1. Overview: 1) The vehicle is tasked with mapping a semantic class (here, *car*) with unknown prior location(s). 2) Using the system described in this paper, the vehicle uses visual and positional information to create a 2.5D semantic map on the fly. 3) Using the map, the vehicle flies towards objects of the class of interest (here, *car*) and acquires high-quality imagery, useful for tasks such as 3D reconstruction. Here we show a model reconstructed from data captured with autonomous scouting.

In support of this goal, this paper proposes a novel *semantic mapping* system to estimate the presence and metric location of the semantic classes of interest (e.g. cars) in its surroundings, so a separate planning system (beyond the scope of this paper) can create information-gathering plans. The map is continuously updated on-the-fly from forward-looking camera imagery and global state estimation, using only on-board computing.

The choice to use a forward-facing (in fact, tilted downwards a 15°) is worth noting. We use this arrangement, as opposed to a downward-facing camera, in order to be able to perceive longer ranges quickly without needing to fly long distances or extremely high altitudes. Unfortunately, this also makes recognition and reconstruction more difficult. Naturally, the choice is not mutually exclusive, and ideally, we would have both.

There are several challenges in the design of such a system. First, the recognition of semantic categories from visual data is highly challenging task. In this case, the difficulty is compounded by the fact that in MAVs with forward-facing cameras, objects will have highly variable appearance as they are captured from different heights and angles. Second, it is non-trivial to reconstruct 3D metric maps from monocular imagery, especially for distant objects and with arbitrary camera motion patterns. Finally, to be useful the system must operate in online and in real time, using the relatively constrained on-board computing on our vehicle.

In order to face these challenges, we make the following contributions:

- We design a custom Deep Learning architecture for 2D semantic segmentation that achieves a good accuracy/speed trade-off for our application. Our starting point are recent convolutional architectures [1], [2], but we empirically make various modifications to optimize for our scenario. In order to train this network, we assembled and labelled a new dataset consisting of oblique aerial imagery gathered from publicly available videos, as well as our own field data.
- We propose a new 2.5D mapping system to efficiently estimate the location of the semantic classes found by the semantic segmentation system. Instead of solving the full 3D reconstruction problem, we assume we have access to a Digital Elevation Map (DEM) of the region, and we project the 2D measurements on this map, while exploiting for available semantic knowledge. DEMs are freely available for many places in the world, including most of the United States. The mapper aggregate measurements over time, making use of knowledge regarding typical heights of objects to improve its accuracy.
- We evaluate each part of our system, and show the integrated system autonomously completing a closed-loop data gathering mission in the field. As part of this evaluation, we have assembled a dataset of labeled aerial imagery.

We will make code and datasets for the semantic segmentation system available at `http://dimatura.net/mavs`.

II. Related Work

A. Semantic Segmentation

Semantic segmentation of RGB imagery is a highly active topic in computer vision. As for most classification tasks in computer vision, the state of the art has been considerably advanced by Deep Learning. In particular, Fully Convolutional Networks [1] achieved high performance and efficiency by adapting networks for single-label prediction to perform dense pixelwise prediction with a single forward pass. Similar models [3]–[5] were proposed at approximately the same time. Since then, most work has focused on optimizing accuracy (e.g. [6], [7]), but relatively little attention has been paid to optimizing computational cost, and in particular, per-image latency (as opposed to memory usage). Recent exceptions include ENet [8] and FastNet [9].

B. Robotic Mapping

Reconstruction of geometric maps from visual and (optionally) inertial sensing data is a well-studied topic in the Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM) literatures.

Algorithmic and computational advances have made it feasible to employ these systems in robotics for real-time decision making; recent relevant examples include SVO [10], which was used for elevation mapping with nadir-looking cameras in a UAV [11].

While this work shows impressive results, it is not applied to frontal-facing cameras, a considerably harder problem, given the relatively small (or non-existent) parallax induced by camera motion in this scenario, especially for distant objects. Recently, Hinzmann et al [12] have proposed a mapping approach optimized for fixed-wing UAVs with oblique cameras. In the future, we will evaluate the incorporation of this kind of approach in our system.

C. Semantic Mapping

Some form of semantic mapping in robotics frequently arises in robotic systems using both semantic and spatial information to navigate; see [13] for a review and taxonomy.

Sengupta et al [14] present an influential system using images and depth to create 3D segmentations for street-level imagery. A more recent, similar approach is [15]. Brostow et al [16], and more recently [17] use monocular imagery for semantic segmentation and 3D reconstruction. Savinov et al [18] jointly use semantic predictions and monocular imagery to create 3D semantic reconstructions. Compared to these approaches, we make various simplifications in order to be able use our system online and onboard the embedded platform on our vehicle.

An impressive recent work is [19], which performs terrain classification with a UAV to support search and rescue missions. Another relevant work is [20], which uses vision to find landing zones. Most of these works use top-down imagery, and it is unclear how their results generalize to oblique imagery. In addition, computation is performed off-board.

In summary, to our knowledge online semantic mapping, on-board an MAV is still an open problem when using oblique monocular imagery.

III. System Overview

The goal of the Semantic Mapping system is to inform the planning system about the presence and approximate location of the classes of interest in its surroundings, so it can create information-gathering plans. It does so by means of a *semantic map*, a metric map that is annotated with localized predictions regarding semantic classes.

Thus, in order to be useful, the system must operate online and in real time, in order to keep the map updated as new sensor data is acquired. Additionally, it must also be capable of recognizing and localizing distant (20 m to 200 m) objects, as its function is primarily to help the vehicle decide where to go, and secondarily to describe where it has been.

To this end, the semantic mapping system must answer two questions about the scene: *what* objects of interest are in it, if any, and *where* are they, in physical space. To answer these questions, our semantic mapping module has two main stages. In the first stage, *semantic segmentation*, we use a deep learning system to label monocular camera imagery. In the second stage, *mapping*, we project the segmentation into a 2.5D grid map which maintains the robot's belief about the semantic class of each grid cell. We describe each stage in further detail in the following sections.

IV. Semantic Segmentation

In the *semantic segmentation* stage, the goal is to assign one of K predefined semantic labels to each pixel in an RGB image. In this paper, the semantic classes are *car* and *background*, where the background class simply corresponds to anything that is not of interest. The choice of semantic classes was driven mainly by pragmatic reasons concerning our testing sites and available data, but the framework extends naturally to arbitrary semantic classes.

Semantic segmentation is closely related to *object detection*, for which the most common goal is to predict a bounding box around each instance of an object class. In this work we prefer the pixel-level semantic segmentation approach over the detection approach, for several reasons: 1) Current algorithms for segmentation are faster, with the possible exception of recent one-shot approaches (e.g., [2]); see [21] for a survey of speed versus accuracy in object detection. 2) We are interested in classes that may not be easily enclosed in a box, such as buildings. 3) We do not require instance-level segmentation; knowing the presence and approximate location of the class of interest suffices. 4) The model is trivially extended to multiple classes. Nonetheless, proposal-based approaches such as Faster RCNN [22], may present advantages for detection of small objects, at some computational expense; this may be an interesting evaluation for future work.

As summarized in section II, in recent years the state of the art has been significantly advanced by Deep Learning, and in particular Fully Convolutional methods [1], which constitute our starting point.

To apply these networks in our project we faced two challenges. First, we found the architectures to be too slow for real-time operation on our embedded platform. Second, we found that off-the-shelf architectures and datasets were optimized for ground-level, prominent objects in the image, whereas we are interested in distant objects that only occupy a few pixels.

Thus, for this project we created a custom architecture and dataset, as we describe below.

1) Architecture: Our main architecture, ScoutNet, is shown in fig. 2. The structure is similar to FCNs [1]. FCNs consist of a Directed Acyclic Graph (DAG) of convolutional and pooling layers, with a 3-channel RGB image as the input and a K-channel "label image" as the output, not necessarily the same size as the input. The network is trained end-to-end by minimizing the pixelwise cross-entropy

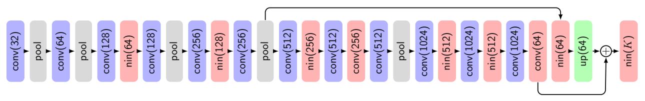


Fig. 2. Our network architecture for a problem, which takes as input an RGB image into the leftmost layer and outputs a K-class image of 1/16 resolution from the rightmost layer. Blue blocks are 3×3 convolutional layers. Gray blocks are pooling layers with two-pixel stride and 2×2 receptive fields. Pink blocks are 1×1 convolutional layers. The green block is a transposed convolutional layer with a half-pixel stride, i.e. it upsamples by a factor of two. The plus symbol corresponds to elementwise addition. All convolutional layers use leaky RelUs and have single stride. Figure best viewed electronically.

loss with stochastic gradient descent. At runtime, inference for pixelwise labelling is purely feed-forward and can be performed efficiently with GPUs.

Most semantic segmentation architectures have focused on maximizing accuracy, at the expense of memory and computing requirements. This becomes evident when applying these methods on relatively low-power platforms such as the NVIDIA TK1, in which inference times for the FCN-VGG16 from [1] proved to be more than a second per image. Therefore, we made various experiments and modifications towards a faster architecture, even at the expense of accuracy.

Our main architecture, which we call ScoutNet, is shown in fig. 2. The initial layers are inspired by the Darknet architecture, which was also used as the basis of the YOLO9000 [23] detection network. Here, we adapt it for segmentation instead. Compared to FCN-VGG16, the number of filters is reduced in the initial layers, and 1×1 convolutional layers are used to further decrease the number of feature maps while retaining useful information. We also eschew the heavy convolutionalized fully-connected layers from the original approach, as in [6], and remove padding, which was observed by [24] to have little effect on accuracy. Like the original FCN, we use skip layers fused by elementwise addition of feature maps, which we observed to be beneficial.

Finally, we also output a lower-resolution labeling. In the FCN, regardless of the effective classification resolution, the output is scaled to the original resolution at the end, even when training. Instead, we simply output the low-resolution output (1/16 of the input, in our case); during training and validation, we downscale the label image. It must be noted that this changes the objective function, as it is possible for small labeled objects to disappear when downscaled.

Our choice was motivated by the observation that most of the information regarding the presence of objects was found already in the lower-resolution layers; the upsampling layers mostly served to add higher-frequency detail, which in our case is not required, as we only need to detect the object presence and approximate location. At the same time, increasing the output resolution through upsampling adds some cost to runtime inference, especially when the operation is not optimized by the GPU backend.

However, while high output resolution is not essential, high *input* resolution is important, since smaller objects (in image space) are harder to detect; for highly downsampled images, many of the smaller objects simply disappear. Hence, we choose to classify images at 896×896 resolution, higher than e.g., the 224×224 resolution commonly used by other architectures since AlexNet [25]. For the output resolution, we use 56×56, i.e., a 16-pixel stride.

In the first generation of our vehicle, which featured an less powerful platform, we used a more lightweight variation of ScoutNet, which we call ScoutNet v0. The network has approximately half the filters for each layer, and due to memory constraints, classification for an 896×896 image is performed by dividing the image into four 448×448 tiles and classifying each separately. This results in border artifacts. In our current vehicle, which features the NVIDIA TX2 platform, we use ScoutNet and classify the whole image at once.

2) Dataset: To address the data issue, we created our own dataset. To reliably detect the classes of interest we need to learn how they appear from the highly varied viewpoints and ranges we encounter in MAV data; but to our knowledge, there is no dataset for object detection or semantic segmentation for oblique, low-altitude (10 m to 40 m) aerial imagery. Instead, existing datasets feature top-down views (e.g., VEDAI [26]) or are biased towards ground-level imagery (e.g., ADE20K [27], Pascal-Context [28]).

Fortunately, thanks to the recent popularity of camera-equipped consumer MAVs, thousands of aerial videos from around the world have been made publicly available on video-sharing websites such as YouTube [29]. These videos vary widely in location, season, time of day, camera intrinsics, video quality, and so on, making for a diverse but challenging source of data.

After downloading hundreds of videos by using various relevant keywords, we performed some elementary analysis to explore the data and filter out unrelated videos. By performing k-means on frame features extracted with a pre-trained network [30], we grouped semantically similar, including clusters of unrelated videos (fig. 3).

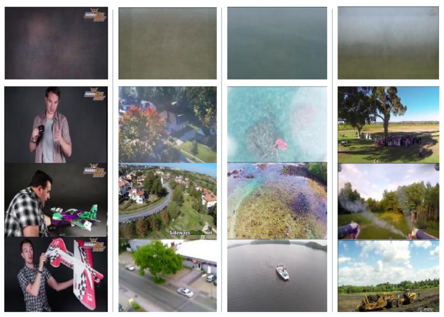


Fig. 3. Frames belonging to videos in automatically extracted clusters, with one cluster in each column, and the top row showing the average cluster frame. The cluster on the left corresponds to videos of reviews for MAV hobbyists, which we discard.

We then manually chose a diverse set of videos and labeled the cars in the dataset with polygons. Our dataset has 1459 images, of which 845 have visible cars, with more than 8000 car instances. We call this the MAVCAR dataset.

Finally, we also created another dataset consisting of 500 images captured from our own field experiments, spanning two years and three locations around western Pennsylvania. Like before, we label cars only. We call this the FIELD dataset.

Example images from the datasets are shown in fig. 4.

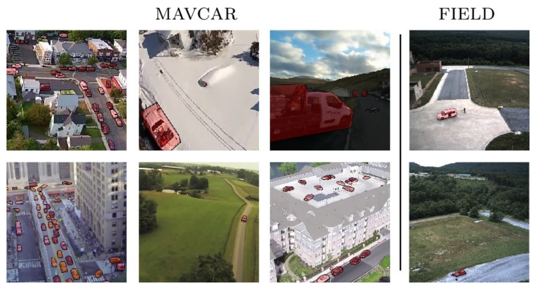


Fig. 4. Example images from our datasets.

V. Mapping

Given a semantically classified image, we want to map the objects in detected in the image in 3D, as well as model regions for which the information in measurements is uncertain. Since this mapping has to be performed on board the vehicle, the driving requirement of the application is computation time. To perform the mapping operation we use the images with soft pixelwise semantic predictions, together with the robot's global pose estimate and a pre-existing Digital Elevation Map (DEM). We exploit prior knowledge of the world (e.g., every object rests on the ground) and use the digital elevation map to infer the 3D structure of the environment.

Given a global pose by state estimation filter, each pixel in the labeled image defines a ray originating at the camera center and passing through the pixel center, providing bearing measurements for semantic objects. Occupancy grid mapping is a basic tool used by mobile robots to represent their beliefs regarding the spatial state of their surroundings when range and bearing measurements are available. The standard algorithm [31] reduces computational complexity by assuming cells in a grid are independent binary random variables and measurements are independent, given a cell's true occupancy value. These assumptions have been shown to work effectively with sensors that provide both range and bearing.

However, a semantically classified image provides bearing only measurements through rays originating from camera pose, making the ray independence assumption limiting. To fully exploit the bearing only measurement and the semantic structure knowledge of the world, we need to model ray dependence. Section V-A and section V-B describe how we model dependence amongst observations while still allowing for an online mapping algorithm.

TABLE I

DATA MEMBERS OF GRID CELL C_{ij} FOR CLASS c.

Symbol	Description
$h_u^{c_{ij}}$	The highest height at which a ray with label c passes over or intersects the cell C_{ij}.
$h_l^{c_{ij}}$	The lowest height at which a ray with label c passes over or intersects the cell C_{ij}.
$n_f^{c_{ij}}$	The number of rays with label c that pass over or intersect the cell C_{ij} at a height less than h_c.
$n_a^{c_{ij}}$	The number of rays with label other than c pass over or intersect the cell C_{ij} at a height less than h_c.
$p_f^{c_{ij}}$	The cumulative probability of rays with label c that pass over or intersect the cell C_{ij} at a height less than h_c.
$n_a^{c_{ij}}$	The cumulative probability of rays with label other than c that pass over or intersect the cell C_{ij} at a height less than h_c.
$l_o^{c_{ij}}$	Integrated logodds of an object of class c being present in the cell C_{ij}.

A. *Exploiting Prior Knowledge*

Assume that objects of our interest, represented by $\mathcal{L}_{\mathcal{M}} = \{c_1, c_2, \ldots, c_n\}$, rest on the ground and we know the likely height $h_{c_i} \forall c_i \in \mathcal{L}_{\mathcal{M}}$. We model the world as a $2.5D$ grid. In every cell, C_{ij} of the grid at location i, j, we store the heights at which rays pass over the cell for all classes by casting rays originating from the classified image, table I. We are interested in finding the cells where the height of rays passing over the cell match the height of object we are looking for, while accounting for occlusions and limited field of view. This leads to following cases for a given class in a cell C_{ij}, see fig. 5 :

Case 0. Average probability of rays that pass over cell C_{ij} with a label other than class c is greater than average probability of rays with class c.

Case 1. Rays of some other class pass from below and above the class of concern over the cell C_{ij}.

Case 2. Rays of some other class pass from below and nothing is observed above the class of concern over the cell C_{ij}.

Case 3. Nothing is observed above or below the class of concern over the cell C_{ij}.

Case 4. Nothing is observed below and some other class is observed above the class of concern over the cell C_{ij}.

Case 1 implies that the cell is well-observed. Therefore, $h_u^{c_{ij}}$ should be close to or greater than h_c and $h_l^{c_{ij}}$ should be close to the ground height. *Case 2* implies that the upper part of the object could not be sensed due sensing geometry or multiple, large semantic objects are present. Hence, $h_l^{c_{ij}}$ should be close to ground. Similarly, *Case 3* implies that there is not enough evidence to confirm or deny a class and *Case 4* implies that $h_u^{c_{ij}}$ should be greater than h_c and $h_l^{c_{ij}}$ should be less than h_c. These cases lead to eq. (1), that is used to determine whether there is positive, negative or lack of evidence in the current classified frame regarding the presence of object of class c over the cell C_{ij}:

$$\phi_{ij}(c) = \begin{cases} 0.5, & \text{Case 3} \\ e^{\alpha_k h_l^{c_{ij}}} e^{\beta_k (h_c - h_u^{c_{ij}})/h_c} \dfrac{p_f^{c_{ij}}}{n_f^{c_{ij}}}, & \text{Case k} \end{cases} \tag{1}$$

Where $k \in [0,4] \setminus 3$ and α_k, β_k are negative constants that enable us to change the weights of the measurements according to the cases encountered. We use the following values for these constants: $\alpha_0 = \beta_0 = -100$, $\alpha_1 = \beta_1 = -10$, $\alpha_2 = -10, \beta_2 = -1$, $\alpha_4 = -1, \beta_4 = -10$. A value of $\phi_{ij}(c)$ close to 0.5 indicates lack of evidence, and $\phi_{ij}(c) < 0.5$ indicates negative evidence and $\phi_{ij}(c) > 0.5$ positive evidence for the presence of class c in cell C_{ij}.

B. *Temporal Evidence Integration*

$\phi_{ij}(c)$, enables the algorithm to model the dependence amongst rays, while treating the cells independently. We assume at any given cell, the log odds of probability of observing a class c is given by a constant γ. Each class in a cell is represented as an independent binary random variable, as a cell can have objects of multiple classes. Once the nature of evidence $(\phi_{ij}(c))$ is identified, log-odds for each class in each cell are updated with eq. (2).

$$l_o^{c_{ij}} = \begin{cases} l_o^{c_{ij}} + \gamma(n_f^{c_{ij}} - n_a^{c_{ij}}), & n_f^{c_{ij}} \geq n_a^{c_{ij}} \wedge \\ & \phi_{ij}(c) - 0.5 \geq \zeta \\ l_o^{c_{ij}} + \gamma(n_f^{c_{ij}} - n_a^{c_{ij}}), & n_a^{c_{ij}} \geq n_f^{c_{ij}} \wedge \\ & 0.5 - \phi_{ij}(c) \geq \zeta \\ l_o^{c_{ij}}, & \text{otherwise} \end{cases} \tag{2}$$

Where ζ is a small positive number less than 0.5. We use $\zeta = 0.2$ and $\gamma = 1$. Each semantically classified image is integrated with the grid and $l_o^{c_{ij}}$ is updated for every cell that needs updating; this process is repeated for every input semantically classified image.

The next section presents the hardware system on which we run the semantic mapping system to enable autonomous scouting. Preliminary results for the mapping algorithm are presented in section VII-B.

VI. PLATFORM

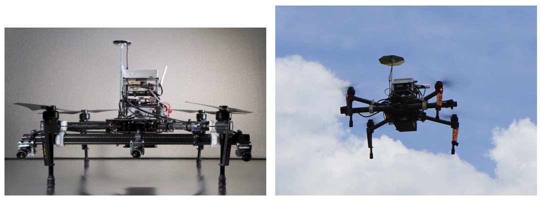
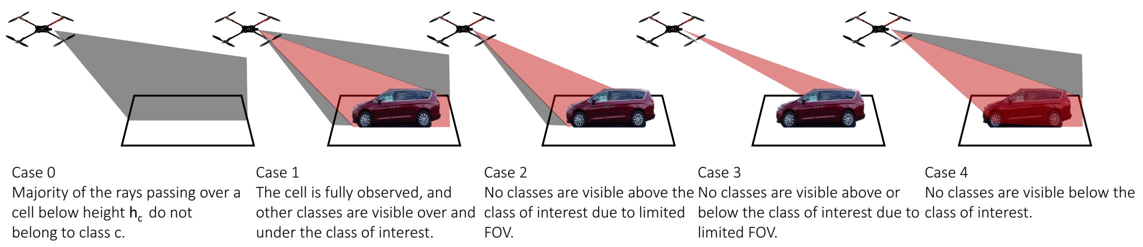


Fig. 6. Aerial Platform.

Our current MAV is depicted in fig. 6. The base platform is an off-the-shelf quadrotor DJI vehicle retrofitted with our own sensors and computing payload designed for autonomous scouting.

1) Sensing: The sensor suite consists of a monochrome stereo camera pair, a monocular color camera, an integrated GPS/INS unit and a barometer. The GPS/INS system and the barometer are used for state estimation.

All cameras are forward-facing, tilted downwards at $15°$, an orientation well suited for low-altitude ($< 40\,\mathrm{m}$) operation. The horizontal field of view for this camera is approximately $60°$, which we considered a good compromise between coverage and object size, given the sensor resolution of 1600×1200 pixels.


Case 0
Majority of the rays passing over a cell below height h_c do not belong to class c.

Case 1
The cell is fully observed, and other classes are visible over and under the class of interest.

Case 2
No classes are visible above the class of interest due to limited FOV.

Case 3
No classes are visible above or below the class of interest due to limited FOV.

Case 4
No classes are visible below the class of interest.

Fig. 5. Illustration of different scenarios that arise due to scene and sensing geometries, when semantically classified images are used to map objects using a camera on an unmanned aerial vehicle.

2) Hardware Platform: All computation for autonomous operation is performed on-board. To this end, we equip the MAV with various embedded computers. In the first generation of our vehicle, we used two embedded ARM computers, an NVIDIA TK1, and an ODroid XU4, to perform perception and planning tasks, respectively. In the second generation we use an NVIDIA TX2 for all computing.

3) Software Platform: Both computers use ROS on Ubuntu Linux. Our segmentation and mapping methods run concurrently as ROS nodes and communicate through messages. The segmentation node, implemented in Python, uses the Theano [32] and Lasagne [33] libraries library with the Nvidia CuDNN backend to make effective use of the GPU. The mapping algorithm is CPU-only and is implemented in C++.

VII. Results

Here, we present results for each of the two main subsystems in isolation and document the integrated system performing a fully autonomous mission in the field.

A. Semantic Segmentation Evaluation

We first study the performance of ScoutNet on the popular Pascal-Context [28] benchmark, with approximately 10k densely labeled images of indoor and outdoor scenes. We use the same protocol as [1], with 59 semantic categories. In table II, we show the results of the FCN-VGG16 16s [1] evaluated at full resolution, FCN-VGG16LR which is the FCN-VGG16s evaluated at 1/16 resolution, and both of our ScoutNets, also evaluated at 1/16 resolution. We also show timing for each network, evaluated on the NVIDIA TK1 and NVIDIA TX2 GPUs. As can be observed, using low resolution has a strong impact on accuracy, as well as a significant impact on timing. We found this to be more pronounced without optimized support from recent versions of the CuDNN backend, as in our first generation vehicle. We also see that the FCN outperforms both of our networks; however, this comes at a high computational cost.

We also evaluate the same approaches on our MAVCAR and FIELD datasets. For MAVCAR, we use a split of 1230 images and a validation set of 229 images; since some videos appear with more than one labeled frame, we ensure there are no videos For some in common between the training and validation sets, to avoid overfitting. For FIELD, we use the whole dataset as validation, and train with the MAVCAR training set. A threshold of 0.5 was used for all results. In

TABLE II
Performance and Timing on Pascal-Context Validation

Arch.	mIoU	Pixel Acc.	TK1 (s)	TX2 (s)
FCN-VGG16 [1]	37.6	66.8	4.336	0.816
FCN-VGG16LR	**33.3**	**26.6**	3.770	0.698
ScoutNet	20.8	17.8	0.452	0.056
ScoutNet v0	20.6	17.7	**0.306**	**0.038**

TABLE III
Performance on MAVCAR and FIELD Validation

	MAVCAR			FIELD		
	IoU	Prec.	Recall	IoU	Prec.	Recall
FCN-VGG16LR	43.4	**77.0**	49.8	**36.6**	78.6	**40.7**
ScoutNet	**44.9**	70.1	**55.5**	34.6	79.4	37.9
ScoutNet v0	28.4	65.0	33.5	29.8	**79.9**	32.2

table III we see that ScoutNet and FCN perform comparably, with ScoutNet and FCN-VGG16LR having a slight lead in MAVCAR and FIELD, respectively. Figure 7 shows some qualitative examples of ScoutNet on MAVCAR and FIELD datasets, highlighting some failure cases.

B. Mapping

In this section we demonstrate the effects of exploiting semantic knowledge and modeling ray dependence qualitatively, while measuring the sensitivity of the mapping algorithm to height inaccuracies in the DEM. Figure 8-4 shows a canonical scenario where a car, more than 50 m away, is detected by the semantic classification algorithm. Exploiting prior knowledge and modeling dependence allows the mapping algorithm to capture the uncertainty about the presence of a car in the cell occluded by the car, fig. 8-1, whereas if we do not reason about ray interdependence, the occluded cell is also inferred to contain cars fig. 8-3. If both the semantic knowledge and ray interdependence are not exploited, then a simple projection of classified image to the DEM leads to an inference that multiple cells are occupied by a car fig. 8-2. Demonstrating that modeling the ray interdependence and exploiting semantic knowledge leads to better mapping of objects and uncertainties. Figure fig. 8 shows that the algorithm's performance deteriorates in presence of height errors in the DEM. Unsurprisingly, the degradation is faster if DEM underestimates the height of the cells due to observation geometry.

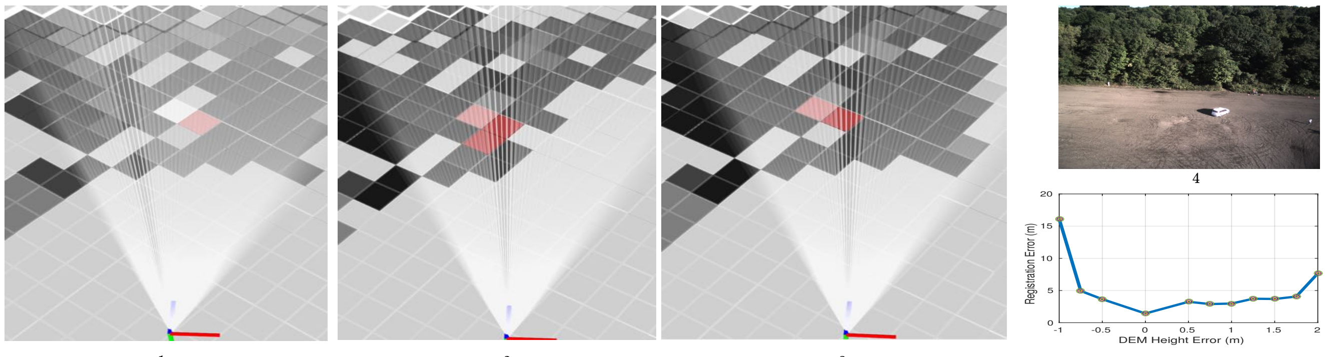


Fig. 8. Figure 1 shows the updated map after a classified image (4) is integrated into our current mapping pipeline; Figure 2 shows the updated map if the classified image is projected on the DEM without exploiting semantic knowledge and Figure 3 shows the updated map if the ray interdependence is not modeled. Dark squares indicate absence of cars and red squares presence of cars. Shades of gray and red signify certainty. Modeling ray interdependence and exploiting semantic knowledge leads to better modeling of uncertainties due to occlusions while providing an improved cell occupancy estimate. Figure 4 provides the sensitivity analysis of mapping performance vs. DEM height errors.

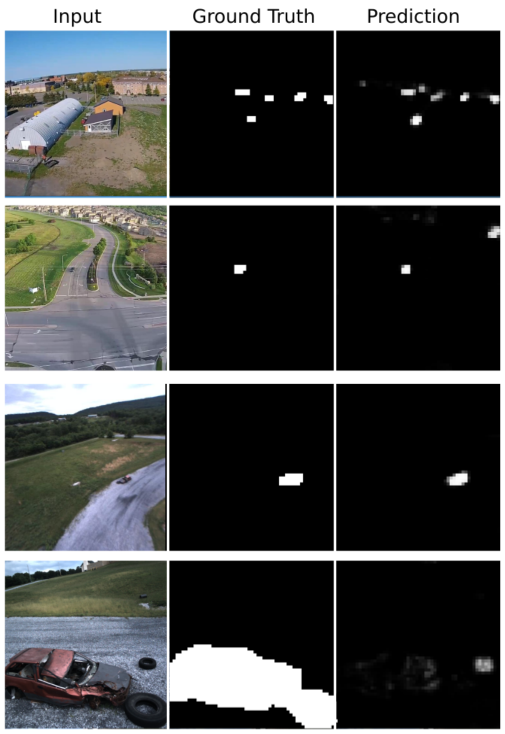


Fig. 7. Examples of ScoutNet, trained on the MAVCAR training set, on MAVCAR (top two rows) and FIELD (bottom two rows) validation images. In each case, the first row shows successful detections, while the second row shows failure cases. For the second row, we can see some false positives: buildings detected as cars in the upper right corner. For the fourth row, we can see our method does not deal well with close-up images of cars.

C. Field Results

We use the semantic mapping pipeline presented in the paper to enable autonomous data gathering missions on-board a custom aerial platform, fig. 6. The vehicle's mission is to scout for cars and collect high-resolution data if a car is found, while making sure it returns to pre-specified location before the battery runs out. Randomized Anytime Orienteering algorithm [34] is used for generating efficient, budgeted data gathering paths for the vehicle, while vision-based obstacle avoidance presented by Dubey et. al. [35] is used for obstacle avoidance. The semantic classification and mapping algorithm is able to detect and map both cars in the environment with sufficient accuracy to enable collection of high resolution data of said cars; see fig. 9.

VIII. Conclusions

In this paper, we have described a semantic mapping system aimed to support autonomous scouting with MAVs. We evaluated the two main components of the system in isolation and demonstrated an integrated autonomous mission.

We are currently improving this system in several ways. We are labeling a larger dataset, including more semantic classes. With this dataset we hope to get a more accurate picture of the performance limits of our method. We are also evaluating methods from recent work in semantic segmentation aimed to optimize performance.

At the same time, we are planning more field experiments, in order to gather data and evaluate the integrated system quantitatively. One interesting improvement would be to generate the observation model for mapping in a data driven fashion. In the future, we are interested in using the image data for dense 3D reconstruction, hoping to avoid the need for an external DEM, or even a GPS.

Acknowledgements

We acknowledge the Office of Naval Research (Grant No. N00014-14-1-06393) and the Qualcomm Innovation Fellowship for providing financial support. We also acknowledge Greg Armstrong for assistance in field experiments and robot construction and repairs.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[2] J. Redmon, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv*, pp. 1–10, 2015.

[3] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture," *arXiv preprint arXiv:1411.4734*, 2014.

[4] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *CoRR*, vol. abs/1505.04366, 2015.

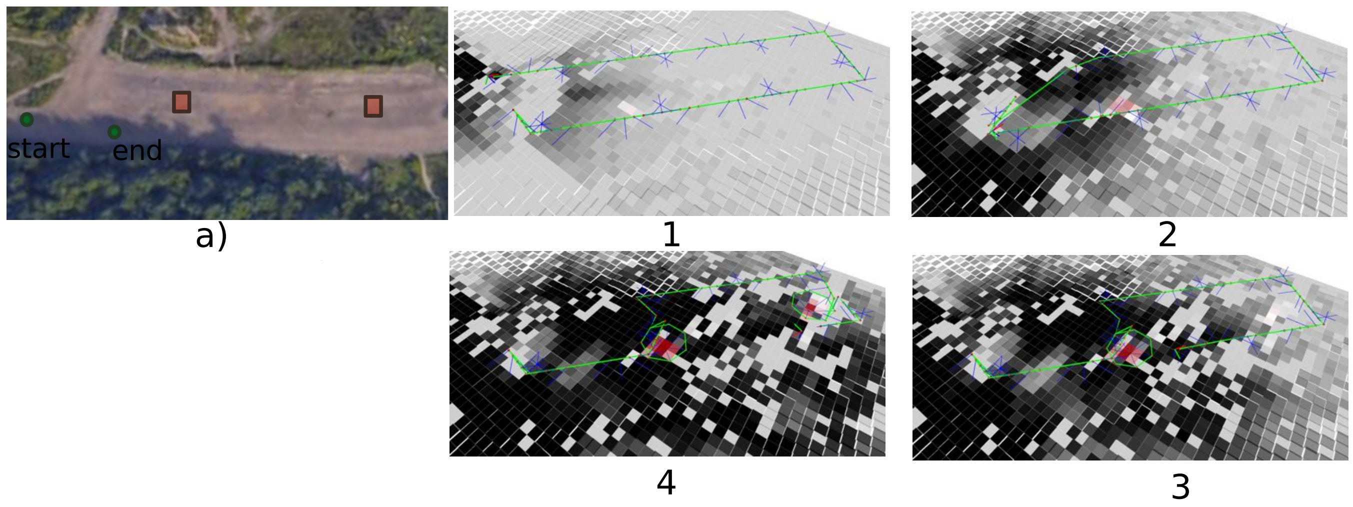


Fig. 9. a) Testing site, start and end are marked by green nodes and car locations are shown in orange. Figures 1,2,3 and 4 show the series of plans at various stages of the exploration mission, Dark squares indicate absence of cars and red squares presence of cars. Shades of gray and red signify certainty. Once the car is recognized, a 360 view of the car is obtained. The mapping pipeline enables detection and data collection for both cars present in the environment.

[5] V. Badrinarayanan, A. Handa, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling," 2015.

[6] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," pp. 1–9, 2015.

[7] Z. Wu, C. Shen, and A. van den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *CoRR*, vol. abs/1611.10080, 2016.

[8] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *CoRR*, vol. abs/1606.02147, 2016.

[9] G. L. Oliveira, W. Burgard, and T. Brox, "Efficient deep models for monocular road segmentation," in *IROS 2016*. IEEE, 2016, pp. 4885–4891.

[10] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, Vision-based Flight and Live Dense 3d Mapping with a Quadrotor Micro Aerial Vehicle," *JFR*, 2015.

[11] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles," in *ICRA*, 2015, pp. 111–118.

[12] T. Hinzmann, T. Schneider, M. Dymczyk, A. Melzer, T. Mantel, R. Siegwart, and I. Gilitschenski, "Robust map generation for fixed-wing uavs with low-cost highly-oblique monocular cameras," in *IROS 2016*. IEEE, 2016, pp. 3261–3268.

[13] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[14] S. Sengupta, P. Sturgess, L. Ladicky, and P. H. S. Torr, "Automatic dense visual semantic mapping from street-level imagery," *IROS*, pp. 857–862, Oct. 2012.

[15] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kähler, D. W. Murray, S. Izadi, P. Pérez, and P. H. S. Torr, "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction," in *ICRA*, May 2015, pp. 75–82.

[16] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, 2008.

[17] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg, "Joint Semantic Segmentation and 3D Reconstruction from Monocular Video," in *ECCV*, 2014, pp. 1–16.

[18] N. Savinov, C. Hane, L. Ladicky, and M. Pollefeys, "Semantic 3d reconstruction with continuous regularization and ray potentials using a visibility consistency constraint," in *CVPR*, 2016, pp. 5460–5469.

[19] J. Delmerico, A. Giusti, E. Mueggler, L. M. Gambardella, and D. Scaramuzza, ""on-the-spot training" for terrain classification in autonomous air-ground collaborative teams," in *ISER*, 2016.

[20] V. R. Desaraju, N. Michael, M. Humenberger, R. Brockers, S. Weiss, J. Nash, and L. Matthies, "Vision-based landing site evaluation and informed optimal trajectory generation toward autonomous rooftop landing," *Autonomous Robots*, pp. 1–19, 2015.

[21] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," pp. 1–10, 2015.

[23] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[24] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *CoRR*, vol. abs/1506.04579, 2015.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[26] S. Razakarivony and F. Jurie, "Discriminative Autoencoders for Small Targets Detection," in *ICPR*. IEEE, 2014, pp. 3528–3533.

[27] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *CVPR*, 2017.

[28] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *CVPR*, 2014.

[29] "Youtube." [Online]. Available: http://youtube.com

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[31] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[32] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *SciPy*, June 2010.

[33] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, *et al.*, "Lasagne: First release." Aug. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878

[34] S. Arora and S. Scherer, "Randomized algorithm for informative path planning with budget constraints," in *ICRA 2017*. IEEE, May 2017.

[35] S. A. Geetesh Dubey and S. Scherer, "Droan - disparity-space representation for obstacle avoidance," in *IROS 2017*. IEEE, September 2017.