SafeUAV: Learning to Estimate Depth and Safe Landing Areas for UAVs from Synthetic Data

Alina Marcu[1,2](✉), Dragoş Costea[2,3], Vlad Licăreţ[2], Mihai Pîrvu[2], Emil Sluşanschi[3], and Marius Leordeanu[1,2,3]

[1] Institute of Mathematics of the Romanian Academy, 21 Calea Grivitei Street, Bucharest, Romania
[2] Autonomous Systems, 7 Iuliu Maniu Street, Bucharest, Romania
{alina.marcu,dragos.costea,vlad.licaret}@autonomous.ro, mihaicristianpirvu@gmail.com
[3] University Politehnica of Bucharest, 313 Splaiul Independenţei, Bucharest, Romania
{emil.slusanschi,marius.leordeanu}@cs.pub.ro

Abstract. The emergence of relatively low cost UAVs has prompted a global concern about the safe operation of such devices. Since most of them can 'autonomously' fly by means of GPS way-points, the lack of a higher logic for emergency scenarios leads to an abundance of incidents involving property or personal injury. In order to tackle this problem, we propose a small, embeddable ConvNet for both depth and safe landing area estimation. Furthermore, since labeled training data in the 3D aerial field is scarce and ground images are unsuitable, we capture a novel synthetic aerial 3D dataset obtained from 3D reconstructions. We use the synthetic data to learn to estimate depth from in-flight images and segment them into 'safe-landing' and 'obstacle' regions. Our experiments demonstrate compelling results in practice on both synthetic data and real RGB drone footage.

Keywords: UAVs · CNNs · Depth estimation · Safe landing

1 Introduction

Nowadays, UAVs have become widely accessible without a substantial investment – anyone can fly a drone, for entertainment or business purposes that cover different domains, such as agriculture or building construction and inspection. Unfortunately, the safety mechanisms embedded on-board are often nonexistent or short-distance oriented, based on ultrasound sensors or depth cameras. Long distance safety abilities, such as ones based on LIDAR, come with an exorbitant price and significant weight for small UAVs [17]. Fortunately, most commercial drones have at least one on-board camera. Recent advances in embedded GPUs enable low power and relatively low cost multi-FPS on-board operation

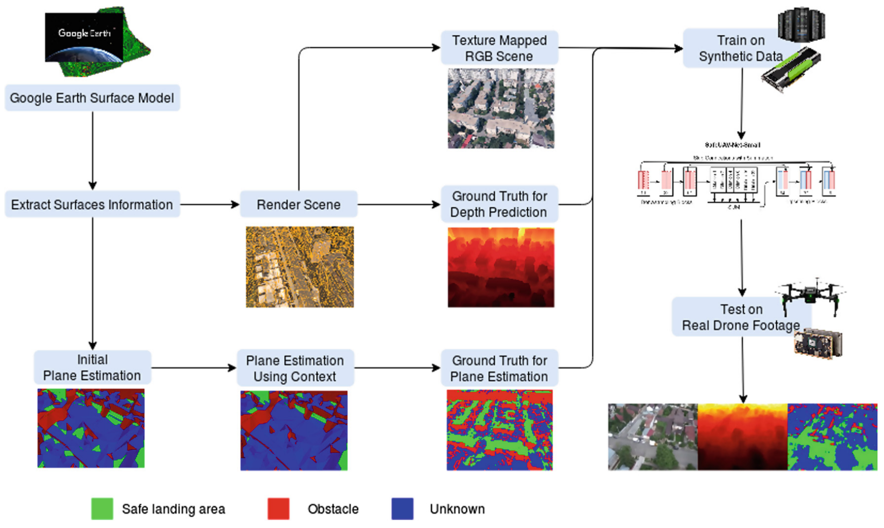
Google Earth Surface Model
Extract Surfaces Information
Render Scene
Texture Mapped RGB Scene
Ground Truth for Depth Prediction
Initial Plane Estimation
Plane Estimation Using Context
Ground Truth for Plane Estimation
Train on Synthetic Data
Test on Real Drone Footage
Safe landing area
Obstacle
Unknown

Fig. 1. Our pipeline for safe landing area and depth estimation. Starting from 3D meshes, we extract RGB, depth, and safe landing labels. We train our proposed CNN on synthetic data and validate our results on drone footage and report beyond real-time results on an embedded device. (Color figure online)

of convolutional neural networks (CNNs) [5], which could make possible high performance computer vision and machine learning approaches.

We address the semantic segmentation of images into 'safe' and potential 'obstacle' regions. We define the problem as classifying surfaces into 'horizontal', 'vertical' and 'other' classes – termed HVO throughout the paper. Potential safe landing areas correspond to the 'horizontal' class, while the 'vertical' and 'other' categories correspond to zones which are likely obstacles or rough landing regions.

Our first contribution is an embeddable CNN for depth, obstacles and safe landing areas estimation. It has comparable performance with state-of-the-art segmentation methods, while being up to 10 times faster and having more than 30 times less parameters.

We also experiment with the idea of processing the input image in two stages. During the first stage we estimate depth and the 'safety' class at the pixel level using a different ConvNet pathway for each task. Since the two tasks are highly interdependent, at the second stage we use the initial outputs of depth and surface class (horizontal, vertical and other), as contextual input to second level ConvNets that can now better re-estimate both depth and the safety category. Thus, we aim to use the depth estimation in order to improve surface class prediction and vice-versa.

The second contribution of our work is the introduction of a synthetic dataset for safe landing, consisting of RGB, depth and horizontal/vertical/other labeled image pairs. The dataset is available online, and can be used to reproduce the results published here as well as improve them with further novelties. We use 3D-reconstructed urban and suburban areas for both automatic ground truth depth extraction and HVO estimation. The pipeline for our approach is shown in Fig. 1.

2 Related Work

Depth Estimation. Ground-level depth estimation is a very active research topic, in terms of learning from unlabeled data [20,21,29], designing different network architectures [18,24,28] and performing domain adaptation [2,16]. Literature for depth estimation from aerial images is very sparse - [14] compares various recent CNN architectures for low-height flight ($<$20 m). There is a very large dataset with 3D information, TorontoCity [27]. Unfortunately it does not include textures for buildings, most of the data being collected only for ground.

Estimating Safe Landing Areas. While most literature focuses on obstacle detection [1], safe landing has been largely unaddressed for embedded use. While some authors have proposed more traditional approaches, such as Naive Bayes classifiers [19], others have tried histogram-based machine learning techniques [3]. No current solution makes use of the power of recent CNN breakthroughs. The most similar to our work proposes a sophisticated pipeline [11], that generates slope, surface normals and terrain roughness, but since it basically reconstructs a 3D model of the ground using bundle adjustment, it takes a couple of seconds on a desktop CPU for a small input image (300×300 pixels) and is unsuitable for on-board processing. In contrast, we learn a fast and relatively small CNN to classify surface normals into three main categories (horizontal, vertical and other), without needing to reconstruct an accurate 3D model.

3 Safe Landing Area Discovery

We aim to discover safe landing areas using RGB signal. We define the problem as a pixel-wise semantic segmentation with 3 classes: *'horizontals'*, *'verticals'* and *'others'*. The horizontals correspond to areas which can be used for automatic landing of UAVs or other aerial vehicles, such as planar ground or rooftops. At the opposite spectrum, the verticals correspond to areas which can be though of as obstacles, such as tall buildings, houses. In general, these are the areas the UAV should stay the farthest off, both during flying as well as landing, to avoid the damage and destruction of the machine. Since the world is not as simple, containing only horizontals and verticals, we define a third class. In this category fall objects such as trees, tilted rooftops, or various irregular shaped objects. These areas can be considered safer for landing than pure verticals, in cases where no horizontals are detected nearby, which can lead to a successful landing in critical situations.

It should be noted that this is an oversimplification of the world, as we assume no semantic knowledge. We are perfectly aware that a horizontal may

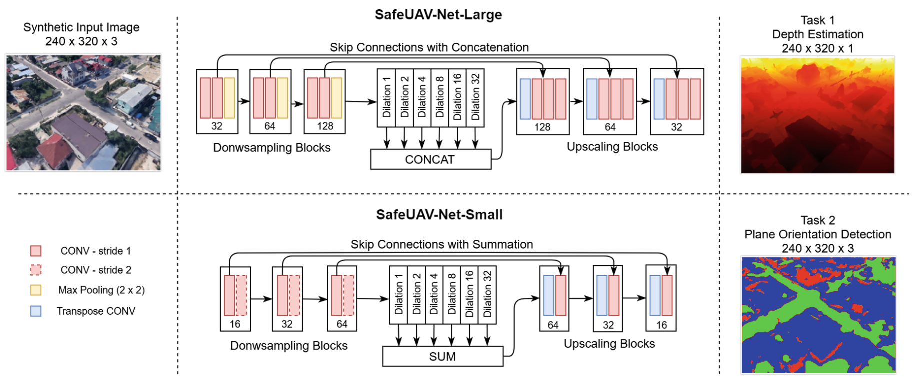

Fig. 2. Our proposed SafeUAV-Nets for both on-board and off-board processing, trained for depth estimation and plane orientation prediction. (Color figure online)

not be semantically safe, such as landing on water or a highway, however we believe that building a low level knowledge of the world, using both depth and HVO, can lead to better results when combined with higher level networks, such as semantic segmentation or object detection, especially since this knowledge is build using only RGB input, in real-time, on low-cost embedded hardware. This is a key objective of this work.

3.1 SafeUAV-Net for Depth and Plane Orientation Estimation

Starting with RGB images, we aim to predict depth and classify plane orientation into three classes: horizontal, vertical and other. Our tasks are strongly related to semantic segmentation as we predict a categorical value for each pixel of the input image. For this purpose, we use a variant of the state-of-the-art U-Net model proposed by [22] for aerial image segmentation. That particular CNN demonstrated state of the art results on various aerial image segmentation tasks, such as the detection of buildings and roads. Compared to previous work, we used concatenation instead of sum for feature aggregation at the U-Net bottleneck phase, in order to capture more information and increase the learning capacity of the network. The role of the concatenation operation in DCNN has been studied in [13] and the predictions produced by these networks were more accurate than the ResNet-like skip connections [10].

Targeting an embedded application, we have developed two variants of the network. The first, named SafeUAV-Net-Large below, runs at 35 FPS on Nvidia's Jetson TX2. The second, named SafeUAV-Net-Small below, is a simplified version of the first, and runs at 130 FPS on an embedded device. For timing details, as well as a comparison with classical architectures, see Sect. 6 and Table 5. The detailed design of our architectures is described in Fig. 2.

SafeUAV-Net-Large is a U-Net-like network [25] with three down-sampling blocks, three up-scaling blocks and a central concatenated dilated convolutions bottleneck, with progressively increasing dilation rates (1, 2, 4, 8, 16 and 32). Each down-sampling block has two convolutional layers with stride 1, followed by a 2×2 max pooling layer. Each up-scaling layer has a transposed convolution layer, a feature map concatenation with the corresponding map from the down-sampling layers and two convolutional layers with stride 1. The number of feature maps are 32-64-128 for the down-sampling blocks and the other way around for the up-sampling ones. All kernels are 3×3. Each dilated convolution outputs a set of 256 activation maps.

SafeUAV-Net-Small follows the similar design principles as the large net - the same number of up-scaling and down-sampling blocks and a central bottleneck with the same dilation rates, but summed features. This way we reduce the computational cost of applying costly convolutional operations over a large set of filters. Each down-sampling block has two convolutional layers, one with stride 1 and the second with stride 2, in order to halve the input resolution. Each up-scaling block has a single convolutional layer and a feature map concatenation with its corresponding map. The number of feature maps are 16-32-64 for the down-sampling blocks and the other way around for the up-sampling blocks. We also reduced the number of filters outputted by each dilated convolution to 128.

Optimization. For the task of depth estimation, we normalize our ground truth labels with values between 0 and 1 and then evaluate the performance of our learning by measuring the L2 loss for our predictions compared to the reference labels. For safe-landing prediction, we used cross-entropy loss for optimizing our models.

Training Details. We used the Pytorch [23] deep-learning framework to train our models. We trained our networks from scratch for 100 epochs each and selected the best epoch, evaluated on our validation set, to report our results. Fine-tuning of models was done for only 50 epochs. We used Adam optimizer, starting with a learning rate of 0.001 for both networks and using a reduce-on-plateau learning rate scheduler, with a factor of 0.1 and a patience of 10.

4 Dataset

4.1 Motivation

While the development of machine learning algorithms for drones and UAVs is an increasingly popular domain, we are aware of no high resolution 3D dataset specifically designed for this purpose. At ground level, there are many datasets and benchmarks such as CityScapes [7] or KITTI [8], useful for training and testing of computer vision applications (for example, in the case of self-driving cars).

Safety laws, airspace regulations, as well as cost and technical reasons prevent easy capturing and creation of a similarly exhaustive dataset from a bird's-eye viewpoint, one that would be better addressed towards UAV machine learning. This is starting to change, thanks to efforts like VisDrone2018 [30] or Okutama-Action [4]. However, most are centred around object detection problems. Those

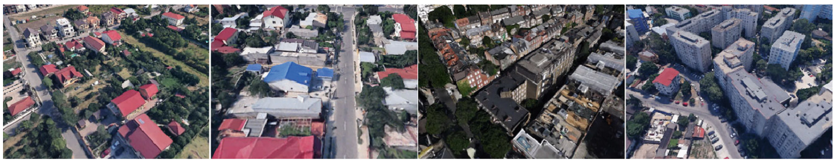

Fig. 3. A sample RGB image from each sub-dataset. From left to right, Suburban A, Suburban B, Urban A and Urban B. (Color figure online)

datasets do not include additional channels apart from RGB and bounding boxes for objects. For the tasks we are tackling in this paper, namely semantic segmentation and depth estimation, we need a large amount of labeled data with pixel-wise annotations for precise predictions.

4.2 Approach

In this paper, we propose a different approach – one that has been similarly gaining traction during the past years: working with computer-generated imagery and labels. Restricting the subject to drones, [26] proposes the generation of a virtual environment using the industry-targeted CityEngine [12] software and produces promising results using real map data as a starting point. CityGML [15] standardizes an exchange format for the storing of 3D models of cities or landscapes, and also offers a basic procedural modelling engine for generating random buildings and cities.

However, the actual output of these systems is not yet realistic; there is no mistaking that this is a virtual world and we therefore believe that the usefulness of these systems for training real-world scenarios is somewhat limited. In contrast, for the purpose of this paper, we chose to construct our virtual dataset with the help and power of the Google Earth [9] application and its real-world derived 3D reconstructions. Up close or from ground level these reconstructions are too coarse to be readily usable, but from a bird's-eye view perspective, several meters above ground, the surfaces and images captured begin to look life-like.

4.3 Dataset Details

We capture a random series of sample images above rectangular patches of ground, with a uniformly randomized elevation between 30 and 90 meters and a stable 45° tilt angle. More precisely, the dataset consists of 11.907 samples in a 80% training (9.524) - 20% validation (2.383) distribution. We further split the data by the type of area they cover, two separate urban areas and two separate suburban ones. A selection of images from each sub-dataset is shown in Fig. 3. We chose these sets as we want to build a robust and diverse dataset that is capable of generalizing over a wide range of environments, while still being relevant to both the depth estimation and plane angle estimation tasks.

We collect two urban areas - Urban A of $3.5\,\text{km}^2$ (3636 samples training, 909 validation) and Urban B covering $\approx 3.3\,\text{km}^2$ (2873 samples training, 719 validation). For the suburban areas, we collect Suburban A of $\approx 1.7\,\text{km}^2$ (1966 samples training, 492 validation) and Suburban B covering $\approx 1.1\,\text{km}^2$ (1049 samples training, 263 validation).

For each of these samples we extract a 640×480 pixels RGB image, an exact depth measure for each pixel, and a semantically labelled image that specifies if a surface is either vertical, horizontal or sloped (other) – HVO estimation.

HVO Extraction Method. We want to generate ground truth label images, but the 3D model we have as a base is still far from perfect, containing many reconstruction errors and stray polygons in otherwise smooth surfaces. We thus approximate plane inclination in three phases[1]:

1. We calculate polygon surface normals and define a maximum error of 10° for horizontals (H) and 20° for verticals (V), while throwing everything else in the sloped bin (O).
2. We pass through the O set and switch polygons to either H or V if (a) their neighbouring polygons in a cubic window are mostly of the same type and (b) the current polygon's surface normal has a more permissive maximum error of 20° for H or 30° for V – we also selectively ignore H surfaces in the V window and vice-versa, as this helps handling 90° corners properly.
3. We similarly pass through the H or V sets and switch polygons to the O set if their neighbouring polygons are mostly sloped. We do this processing because, as previously mentioned, the actual surface model is a noisy one and we wish to identify large patches that are consistently straight (H or V) while ignoring everything that we are either unsure of, either represents a true complex structure that is difficult to classify (O).

Having all our polygons now labeled, we color-code each class and generate a second surface model having the same geometry as the first, but whose face textures now represent semantic information. We can now capture all the images we need, using exactly the same viewpoint for each image in our sample. Lastly, as images from this second model are relatively coarse due to the hard cuts between polygons, we apply a simple post-processing step for smoothing them. We relabel each pixel to have the class that is dominant in a 5×5 pixels window.

Dataset Realism and Quality. The dataset consists of 3D reconstructions from real RGB images. However, due to the ill-posed algorithmic problem of reconstruction, limited resolution and sampling distance, the quality of the reconstructions is not ideal - almost no building facade is vertical, the texture mapping fails to accurately match object geometry and overall, clustering verticals and horizontal regions is a task prone to error. Although the dataset should be called semi-synthetic, due to significant discrepancies between reality and the generated 3D meshes (see Fig. 4), we generally call it synthetic in the paper.

[1] Code and dataset available at https://sites.google.com/site/aerialimageunderstanding/safeuav-learning-to-estimate-depth-and-safe-landing-areas-for-uavs.

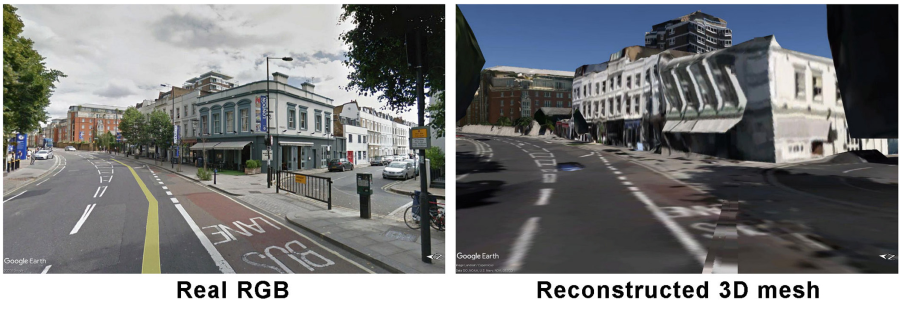

Fig. 4. A sample of RGB vs reconstructed 3D meshes (used for training) from the same location. Note the significant geometric, texture and illumination inconsistencies introduced by the reconstruction in contrast to a real-world capture. The darker regions in the right and left are tree meshes. While these meshes are very difficult to use for learning something at the ground level, from a birds's eye viewpoint they look similar to the real RGB image. (Color figure online)

5 Experiments

We report qualitative and quantitative results on depth estimation and HVO segmentation on all four regions from our dataset.

The metrics used for the HVO task are the network Loss (cross entropy), Accuracy, Precision, Recall and mean intersection over union (mIoU). All the reported values are computed on the unseen validation set. These metrics can be defined in terms of true positives, true negatives, false positives and false negatives, as follows: $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$ and $mIoU = \frac{TP}{TP+FP+FN}$.

The cross-entropy used for HVO can be expressed as:

$$L(y,t) = -\frac{1}{N}\sum_{i,j}\sum_{c=\{H,V,O\}} t_{i,j}^{(c)} * log(y_{i,j}^{(c)})$$

Each prediction $y_{i,j}$ is a probability vector, that corresponds to a confidence level for each of the 3 classes, while the target vector is a one-hot encoded vector, with a value of 1 for the correct class and a value of 0 for the other two.

The metrics used for the depth prediction are the network Loss (Sum of squared error), the Root Mean Squared Error (RMSE) as well as an absolute error (in meters). The SSE Loss function can be expressed as:

$$L(y,t) = \sum_{i,j}(y_{i,j} - t_{i,j})^2$$

The meters error is the summed error in meters over the entire image, scaled by the corresponding dataset-dependant scaling factor (F). This can be expressed mathematically as:

$$L_{met}(y,t) = \frac{1}{N}\sum_{i,j}|y_{i,j} * F - t_{i,j} * F| = \frac{1}{N}\sum(|y_{i,j} - t_{i,j}| * F)$$

The first term is exactly the absolute (L1) loss, while the scaling factor is required due to the rendering process, which is different for all the considered 3D models. $t_{i,j}$ is the ground truth distance and $y_{i,j}$ the predicted distance at the (i, j) location, while N stands for the number of pixels in the current image.

This factor is a constant number that is different for each region, based on how the 3D model is defined, but we normalized all the data to the same values (multiplying the given depth with its factor). This allowed us to train the network on all the 4 regions using the same scale throughout all the data.

For a fair comparison, we also train the two tasks on two standard architectures, the U-net [25] and the DeepLabV3+ [6]. In order to keep the same settings, the input of the network is changed from 240×320 to the default value for each architecture: 572×572 inputs with 388×388 outputs for U-net and 512×512 inputs with 512×512 outputs for the DeepLabV3+ model. This makes some unnormed losses incomparable (such as the Depth L2 loss).

5.1 Qualitative and Quantitative Evaluation

Figures 5 and 6 show RGB imagery and ground truth for both depth and HVO tasks, and also qualitative results on the HVO task (Fig. 5) and depth estimation (Fig. 6) on all four regions from our dataset.

Buildings in suburban regions are significantly sparser than urban regions - continuous building regions occupy large amounts of the urban landscape. The evaluation was done by combining all 4 areas into a single dataset. Furthermore, each urban and suburban area was tested independently, using only the data from its own region, split into train and validation. We report results for each task - depth and HVO.

Table 1. HVO prediction results for SafeUAV-Net-Large and SafeUAV-Net-Small trained on full dataset. HVO_{nn} is the prediction produced by our network after training it with the ground truth HVO_{gt}.

Model	Input	Accuracy	Precision	Recall	mIoU
U-net [25]	RGB	0.729	0.560	0.505	0.356
DeepLabv3+ [6]	RGB	0.840	0.753	0.739	0.597
Small	RGB	0.823	0.728	0.693	0.551
Large	RGB	**0.846**	**0.761**	**0.748**	**0.607**
Small	RGB + $Depth_{nn}$	0.823	0.728	0.696	0.552
Small	RGB + $Depth_{gt}$	0.902*	0.845*	0.840*	0.732*
Large	RGB + $Depth_{nn}$	0.834	0.741	0.726	0.582
Large	RGB + $Depth_{gt}$	0.909*	0.858*	0.848*	0.748*

The inputs noted with *gt* (ground truth) represent the automatically extracted label. Inputs noted with *nn* (neural network output, iteration 1), represent the labels extracted by the first network (which in turn was trained with ground truth).
*Since these networks are trained using ground truth labels, and cannot be used for real-world prediction, they are only presented as an upper bound for each model.

Table 2. Results on depth estimation for SafeUAV-Net-Large and SafeUAV-Net-Small trained on full dataset. Errors are expressed in meters.

Model	Input	RMSE	Meters
U-net [25]	RGB	0.041	9.63
DeepLabv3+ [6]	RGB	0.034	8.49
Small	RGB	0.031	7.22
Large	RGB	**0.026**	**6.09**
Small	RGB + HVO_{nn}	0.037	8.76
Large	RGB + HVO_{nn}	0.027	6.34

As previously stated, the input and output shapes are not identical for all models, so the Loss iteself should not be compared, but rather the RMSE and meters metrics.
We also observe that using RGB + HVO as input, be it ground-truth or a prediction of the network gives a much lower improvement, and even hinders the results in some cases. This comes in contrast to the HVO estimation task, where adding the depth signal improves the results almost every time.

The HVO results (Table 1) show a clear advantage of our SafeUAV-Net-Large over the state-of-the-art CNN. Furthermore, the numbers for our SafeUAV-Net-Small are similar to the large one, even though it has much less parameters. We aimed to further improve safe landing detection using depth labels. Unfortunately, except the small improvement for our small network, the results degraded. We argue this is due to the noise in the depth labeling. To confirm this, we trained with the ground truth depth, achieving the best results. We believe improving the ground truth depth labels could also improve safe landing detection. We provide detailed results for each sub-dataset in Table 3. Training on all datasets results in a more robust overall detection, as intended when selecting the interest regions.

Table 3. Quantitative results of SafeUAV-Net-Large and SafeUAV-Net-Small for the HVO task. We report mean values for Accuracy, Precision, Recall and IoU on the validation sets. The networks in this table were trained using only RGB input. The second part of the table presents the results, using the best model trained on the entire network, comparing the impact of fine-tuning on a specific area.

All	Model	Accuracy		Precision		Recall		mIoU	
	Small	0.81 ± 0.01		0.72 ± 0.01		0.68 ± 0.01		0.53 ± 0.01	
	Large	0.83 ± 0.01		0.74 ± 0.01		0.71 ± 0.02		0.57 ± 0.02	
		Base	Fine-tuned	Base	Fine-tuned	Base	Fine-tuned	Base	Fine-tuned
SubA	Small	0.839	0.841	0.730	0.736	0.671	0.673	0.543	0.546
	Large	0.855	0.865	0.755	0.779	0.723	0.730	0.590	0.609
SubB	Small	0.816	0.819	0.695	0.704	0.598	0.600	0.472	0.475
	Large	0.829	0.845	0.707	0.749	0.671	0.672	0.529	0.549
UrbA	Small	0.810	0.808	0.716	0.714	0.698	0.696	0.547	0.547
	Large	0.844	0.854	0.765	0.784	0.762	0.772	0.618	0.638
UrbB	Small	0.833	0.835	0.760	0.761	0.743	0.748	0.599	0.604
	Large	0.851	0.860	0.782	0.796	0.779	0.789	0.639	0.656

Table 4. Quantitative results of SafeUAV-Net-Large and SafeUAV-Net-Small for the depth estimation task. We report mean values for RMSE as well as absolute error (in meters) on the validation sets. The networks in this table were also trained only using RGB input, and the second part of the table presents the results, with and without fine-tuning the networks.

All	Model	RMSE		Meters	
	Small	0.045 ± 0.009		10.51 ± 2.24	
	Large	0.036 ± 0.005		8.29 ± 1.16	
		Base	Fine-tuned	Base	Fine-tuned
SubA	Small	0.025	0.023	5.59	5.35
	Large	0.022	0.020	4.96	4.67
SubB	Small	0.023	0.019	5.51	4.46
	Large	0.019	0.017	4.72	4.05
UrbA	Small	0.035	0.035	8.39	8.23
	Large	0.031	0.031	7.08	7.02
UrbB	Small	0.032	0.031	7.47	7.17
	Large	0.027	0.026	6.10	5.92

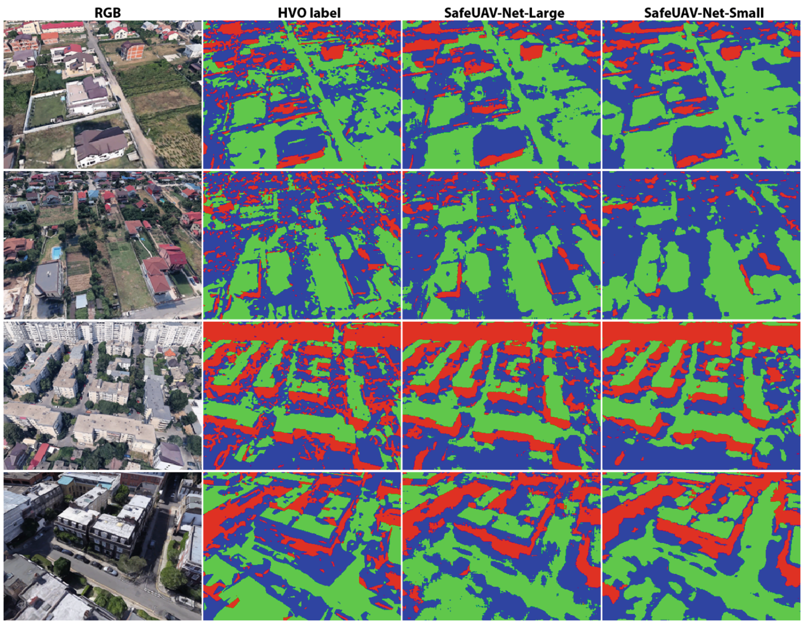

Fig. 5. HVO qualitative results on testing samples from all datasets from SafeNet-UAV-Large. Red stands for horizontal, yellow for vertical and blue for other areas. From top to bottom, SuburbanA, UrbanA, UrbanB and SuburbanB, at various altitudes. The HVO prediction tends to be less noisier and closer to the real 'ground truth'. The performance hit from the smaller network is difficult to notice. (Color figure online)

The depth results are shown in Table 2. This time, both flavours of our proposed architectures outperform established CNNs. Results for each sub-dataset are shown in Table 4. The noisiness of the input is reflected in the higher standard deviation compared to the safe landing task, noticeable on all regions.

We also show results from RGB drone footage in Fig. 7, for both sizes of the network and tasks (depth and HVO). We notice similar performance compared to the synthetic dataset. We believe increasing the size of the dataset could further improve the results.

6 SafeUAV Timings

As described in Sect. 3.1, we develop two versions of SafeUAV-Net - a large one and a small one. We report timings in Table 5 for both desktop GPUs (NVIDIA Tesla P100) and embedded devices (NVIDIA Jetson TX2). While maintaining comparable performance, the small one is suitable for embedded usage.

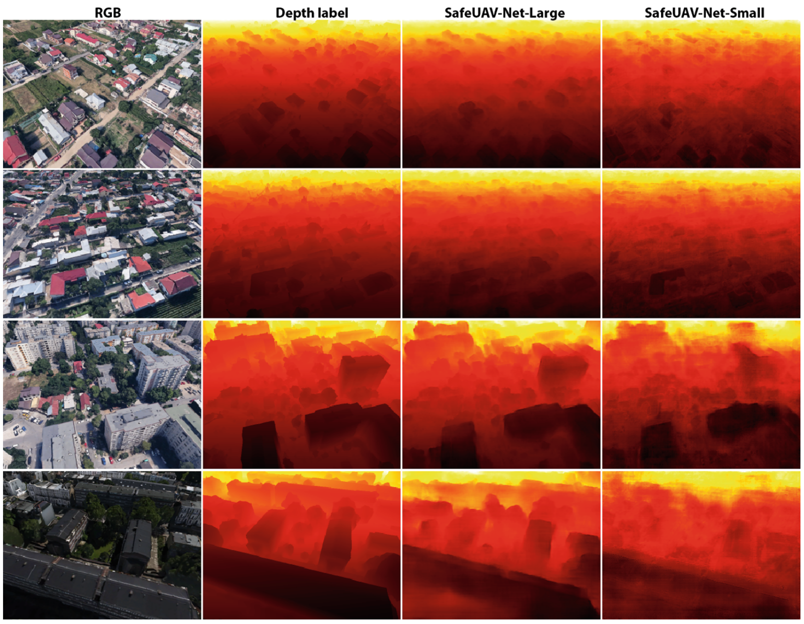

Fig. 6. Depth qualitative results on testing samples from all datasets from SafeNet-UAV-Large. Depth is normalized between 0 and 1 using the hot colormap (0 means close, 1 faraway). From top to bottom, SuburbanA, UrbanA, UrbanB and SuburbanB, at various altitudes. The prediction tends to blur object edges. This is even more visible in the predictions of the smaller network. (Color figure online)

Table 5. Number of parameters (RGB input, 1-channel depth map output), memory usage (inference mode, batch of 1) and inference time for multiple batch sizes for the Safe-UAV networks, compared to the more standard U-net and state-of the art DeepLabV3+.

Network	Number of parameters	Memory usage	Batch size	Tesla P100 (images/s)	Jetson TX2 (images/s)
U-net [25]	31M	1.7 GB	1	502	37
			5	2311	–
			10	3980	–
			20	–	–
DeepLabv3+ [6]	55M	1.9 GB	1	63	–
			5	326	–
			10	611	–
			20	–	–
SafeUAV-Net-Large	24M	927 MB	1	463	35
			5	2333	181
			10	4557	428
			20	7978	–
SafeUAV-Net-Small	1M	433 MB	1	577	138
			5	2871	635
			10	5774	1045
			20	11480	1939

*Dashed values appear because of out of memory errors with the respective boards.

7 Future Work

We aim to use the spatial and temporal continuity present in video sequences in order to generate a more robust prediction. At lower speeds, similar frames could be used to vote on the same region in order to get a better precision at landing time. Additionally, the mesh captured during flight (extracted from RGB used as a texture for the depth map) could be used to improve the surface classification of the same location. We also aim to increase the spatial resolution of the ground truth and efficiently apply the proposed network only on the region of the image that is significantly different from the previous frame.

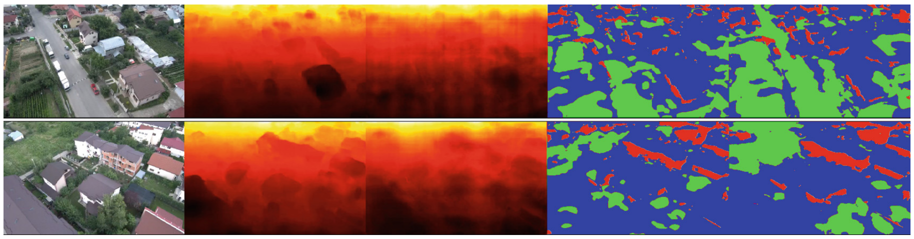

Fig. 7. Qualitative results on real UAV footage. RGB, depth predicted with SafeUAVNet-Small and Large, HVO predicted with SafeUAVNet-Small and Large. We obtain similar results to the synthetic dataset ones. (Color figure online)

Finally, we aim to improve the safety by visual geolocalization - include additional common classes easily derived from HVO (such as buildings and tall structures) and use several discrete geo-localized items to compute an approximate location, given an initial start position. We believe this will improve safe area estimation by conditioning on the predicted geo-location. Automatic visual geolocalization, a task related to recent work [22], could also handle cases of radio signal loss and improve overall UAV navigation safety and robustness.

8 Conclusions

We propose SafeUAV-Net - an embeddable-hardware compatible system based on deep convolutional networks, designed for depth and safe landing area estimation using only the RGB input. Furthermore, we produce and train on a synthetic dataset and show compelling performance on real drone footage. Our extensive experiments on unseen synthetic test cases, where ground truth information is available, show that our system is numerically accurate, while also being fast on an embedded GPU running at 35 FPS (SafeUAV-Net-Large) and 138 FPS (SafeUAV-Net-Small). We believe that the use of our approach on commercial drones could improve flight safety in urban or suburban areas at high speeds and complement the limited range of on-board sensors.

Acknowledgements. This work was supported in part by Romanian Ministry of European Funds, project IAVPLN POC-A1.2.1D-2015-P39-287 and UEFISCDI, under projects PN-III-P4-ID-ERC-2016-0007 and PN-III-P1-1.2-PCCDI-2017-0734. We would also like to express our gratitude to Aurelian Marcu and The Center for Advanced Laser Technologies for providing use GPU training resources.

References

1. Aguilar, W.G., Casaliglla, V.P., Pólit, J.L.: Obstacle avoidance based-visual navigation for micro aerial vehicles. Electronics **6**(1), 10 (2017)
2. Atapour-Abarghouei, A., Breckon, T.P.: Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 18, p. 1 (2018)
3. Aziz, S., Faheem, R.M., Bashir, M., Khalid, A., Yasin, A.: Unmanned aerial vehicle emergency landing site identification system using machine vision. J. Image Graph. **4**(1), 36–41 (2016)
4. Barekatain, M., et al.: Okutama-action: an aerial view video dataset for concurrent human action detection. In: 1st Joint BMTT-PETS Workshop on Tracking and Surveillance, CVPR, pp. 1–8 (2017)
5. Canziani, A., Paszke, A., Culurciello, E.: An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678 (2016)
6. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:1802.02611 (2018)
7. Cordts, M., et al.: The CityScapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213–3223 (2016)
8. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)
9. Google: Google Earth, version 7.3.0 (2018). https://www.google.com/earth/
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
11. Hinzmann, T., Stastny, T., Lerma, C.C., Siegwart, R., Gilitschenski, I.: Free LSD: prior-free visual landing site detection for autonomous planes. IEEE Robot. Autom. Lett. **3**, 2545–2552 (2018)
12. Hu, X., Liu, X., He, Z., Zhang, J.: Batch modeling of 3D city based on ESRI CityEngine (2013)
13. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. 3 (2017)
14. Julian, K., Mern, J., Tompa, R.: UAV depth perception from visual, images using a deep convolutional neural network (2017). http://cs231n.stanford.edu/reports/2017/pdfs/200.pdf
15. Kolbe, T.H., Gröger, G., Plümer, L.: CityGML: interoperable access to 3D city models. In: van Oosterom, P., Zlatanova, S., Fendel, E.M. (eds.) Geo-information for disaster management, pp. 883–899. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-27468-5_63

16. Kundu, J.N., Uppala, P.K., Pahuja, A., Babu, R.V.: Adadepth: unsupervised content congruent adaptation for depth estimation. arXiv preprint arXiv:1803.01599 (2018)
17. Leaverton, G.T.: Generation drone: the future of utility O&M. In: Electrical Transmission and Substation Structures 2015, pp. 190–201 (2015)
18. Lee, J.H., Heo, M., Kim, K.R., Kim, C.S.: Single-image depth estimation based on fourier domain analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 330–339 (2018)
19. Li, X.: A software scheme for UAV's safe landing area discovery. AASRI Procedia **4**, 230–235 (2013)
20. Li, Z., Snavely, N.: MegaDepth: learning single-view depth prediction from internet photos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2041–2050 (2018)
21. Mahjourian, R., Wicke, M., Angelova, A.: Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5667–5675 (2018)
22. Marcu, A., Costea, D., Slusanschi, E., Leordeanu, M.: A multi-stage multi-task neural network for aerial scene interpretation and geolocalization. arXiv preprint arXiv:1804.01322 (2018)
23. Paszke, A., et al.: Automatic differentiation in pytorch. In: NIPS-W (2017)
24. Qi, X., Liao, R., Liu, Z., Urtasun, R., Jia, J.: GeoNet: geometric neural network for joint depth and surface normal estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 283–291 (2018)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. arXiv preprint arXiv:1505.04597 (2015)
26. Tian, Y., Li, X., Wang, K., Wang, F.Y.: Training and testing object detectors with virtual images. IEEE/CAA J. Automatica Sin. **5**(2), 539–546 (2018)
27. Wang, S., et al.: Torontocity: seeing the world with a million eyes. arXiv preprint arXiv:1612.00423 (2016)
28. Xu, D., Ouyang, W., Wang, X., Sebe, N.: Pad-net: multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. arXiv preprint arXiv:1805.04409 (2018)
29. Zhan, H., Garg, R., Weerasekera, C.S., Li, K., Agarwal, H., Reid, I.: Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 340–349 (2018)
30. Zhu, P., Wen, L., Bian, X., Ling, H., Hu, Q.: Vision meets drones: a challenge. arXiv preprint arXiv:1804.07437 (2018)