J-MOD^2: Joint Monocular Obstacle Detection and Depth Estimation

Michele Mancini[1], Gabriele Costante[1], Paolo Valigi[1] and Thomas A. Ciarfuglia[1]

***Abstract*—In this work, we propose an end-to-end deep architecture that jointly learns to detect obstacles and estimate their depth for MAV flight applications. Most of the existing approaches rely either on Visual SLAM systems or on depth estimation models to build 3D maps and detect obstacles. However, for the task of avoiding obstacles this level of complexity is not required. Recent works have proposed multi task architectures to perform both scene understanding and depth estimation. We follow their path and propose a specific architecture to jointly estimate depth and obstacles, without the need to compute a global map, but maintaining compatibility with a global SLAM system if needed. The network architecture is devised to jointly exploit the information learned from the obstacle detection task, which produces reliable bounding boxes, and the depth estimation one, increasing the robustness of both to scenario changes. We call this architecture J-MOD^2. We test the effectiveness of our approach with experiments on sequences with different appearance and focal lengths and compare it to SotA multi task methods that perform both semantic segmentation and depth estimation. In addition, we show the integration in a full system using a set of simulated navigation experiments where a MAV explores an unknown scenario and plans safe trajectories by using our detection model.**

***Index Terms*—Range Sensing, Visual Learning, Visual-Based Navigation**

I. Introduction

OBSTACLE avoidance has been deeply studied in robotics due to its crucial role for vehicle navigation. Recently, the demand for faster and more precise Micro Aerial Vehicle (MAV) platforms has put even more attention on it. To safely execute aggressive maneuvers in unknown scenarios, the MAVs need a robust obstacle detection procedure.

Most fruitful approaches rely on range sensors, such as laser-scanner, stereo cameras or RGB-D cameras [1], [2], [3] to build 3D maps and compute obstacle-free trajectories. However, their use results in an increased weight and power consumption, which is unfeasible for small MAVs. Furthermore, their sensing range is either limited by device characteristics (RGB-D and lasers) or by camera baselines (stereo cameras).

Manuscript received: September, 10, 2017; Revised November 8, 2017; Accepted January, 10, 2018.

*This paper was recommended for publication by Editor Cyrill Stachniss upon evaluation of the Associate Editor and Reviewers' comments. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

[1]All the authors are with the Department of Engineering, University of Perugia, via Duranti 93, Perugia Italy

{thomas.ciarfuglia, paolo.valigi, gabriele.costante, michele.mancini}@unipg.it

Digital Object Identifier (DOI): see top of this page.

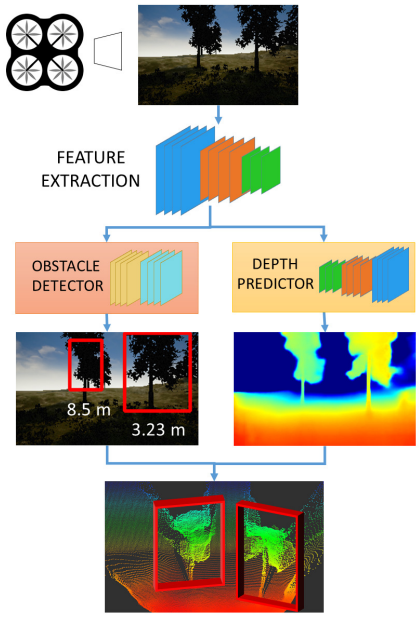


Fig. 1: Overview of the proposed system: the architecture is composed by two networks that perform different, but connected tasks: obstacle detection and pixel-wise depth estimation. The two task are jointly learned and the feature extraction layers are in common. Thus, the resulting model has increased accuracy in depth prediction because of the semantic information received from the detector. On the other hand, the detector learns a better representation of obstacles through depth estimation.

Monocular Visual SLAM (VSLAM) approaches address the above limitations by exploiting single camera pose estimation and 3D map reconstruction [4], [5], [6], [7]. Nevertheless, these advantages come with costs: the absolute scale is not observable (which easily results in wrong obstacle distance estimations); they fail to compute reliable 3D maps on low-textured environments; the 3D map updates are slow with respect to real-time requirements of fast manoeuvres. With careful tuning, these approaches can be used for obstacle avoidance.

At the same time there are other approaches that tackle the problem more specifically. In this respect, a step toward more robust obstacle detection has been made by monocular depth estimation methods based on Convolutional Neural Networks (CNNs) [8], [9], [10]. Compared to standard VSLAM strategies, these works train CNN-based model to quickly compute depth maps from single image, which allows for fast trajectory replanning. However, as any data-driven approach, these depth models are biased with respect to appearance domains and camera intrinsics. Most of the CNN architectures

so far proposed address the more general task of pixel-wise depth prediction and are not specifically devised for obstacle detection. However, recent works [11] [12] have digressed from this trail, proposing multi task network architectures to jointly learning depth and some semantic property of the images. These works show that the mutual information is beneficial to both tasks.

Driven by the previous considerations, in this work we propose a novel CNN architecture that jointly learns the task of depth estimation and obstacle detection. We aim to get, at the same time, the detection speed of CNNs approaches and more robustness to scale and appearance changes, using the joint learning of the depth distribution.

The combination of these two tasks gives them mutual advantages: the depth prediction branch is informed with object structures, which result in more robust estimations. On the other hand, the obstacle detection model exploits the depth information to predict obstacle distance and bounding boxes more precisely. Our approach is similar to [11] and [12], but is specifically devised for obstacle detection, and not generic scene understanding, in order to achieve more robustness to appearance changes. We show the comparison with these two aforementioned methods in the experimental part of the work. We demonstrate the detection and depth estimation effectiveness of our approach in both publicly available and brand new sequences. In these experiments, we prove the robustness of the learned models in test scenarios that differ from the training ones with respect to focal length and appearance. In addition, to demonstrate the detection advantages of the proposed detection system, we set up a full navigation avoidance system in a simulated environment with a MAV that detects obstacles and computes free trajectories as it explores the scene.

II. Related Work

The most straight-forward approaches to obstacle detection and depth estimation involve RGB-D or stereo cameras.

Unfortunately, these sensors suffer from limited range, in particular stereo systems, that require large baselines to achieve acceptable performances [13].

For example, some authors explored push-broom stereo systems on fixed-wing, high speed MAVs [14]. However, these approaches require too large baselines for small rotary wing MAVs. In addition, while short-range estimations still allows safe collision avoidance, it sets an upper bound to the robot's maximum operative speed.

For all these reasons the study of alternative systems based on monocular cameras becomes relevant. Even with the limitation of monocular vision, our method can detect and localize obstacles up to 20 meters and compute dense depth maps up to 40 meters with a minor payload and space consumption.

Monocular obstacle detection can be achieved by dense 3D map reconstruction via SLAM or Structure from Motion (SFM) based procedures [6], [15], [16]. These systems perform a much more complex task though, and usually fail at high speeds, since they reconstruct the environment from frame to frame triangulation.

In addition, with standard geometric monocular systems it is not possible to recover the absolute scale of the objects, without using additional information. In [17] the scale is recovered using the knowledge of the camera height from the ground plane, while [18] uses a inference based method on the average size of objects that frequently appear in the images (e.g. cars), then optimize to the whole trajectory. The lack of knowledge of the scale makes the obstacle avoidance a difficult task. For this reason, some approaches exploit optical information to detect proximity of obstacles from camera, or, similarly, detect traversable space, or use hand-crafted image features [19], [20], [21], [22], [23].

However, recently proposed deep learning-based solutions have shown robustness to the aforementioned issues. These models produce a dense 3D representation of the environment from a single image, exploiting the knowledge acquired through training on large labeled datasets, both real-world and synthetic [24], [8], [25], [9].

A few of these methods have been recently tested in obstacle detection and autonomous flight applications. In [26], the authors fine-tune on a self-collected dataset the depth estimation model proposed by [24] and use it for path planning. In [10] the authors exploit depth and normals estimations of a deep model presented in [8] as an intermediate step to train an visual reactive obstacle avoidance system. More recently, [10] proposed a similar approach, regressing avoidance paths directly from monocular 3D depth maps.

However, the aforementioned methods solve the task of depth estimation and from it derive the obstacle map. Another set of approaches use semantic knowledge to strengthen the detection task. On this line the works of [27], [11] and [12] train a multi task architecture for semantic scene understanding that is reinforced by the joint learning of a depth estimation task. However, these methods show better performances on classes such as "ground" or "sky". Our intuition is that current depth estimators overfit their predictions on these classes, as they tend to have more regular texture and geometric structures.

On the contrary, in robotic applications we want to train detection models to be as accurate as possible when estimating obstacle distances.

Following this multi task approaches, we propose a novel solution to the problem by jointly training a model for depth estimation and obstacle detection. While each task's output comes from independent branches of the network, feature extraction from their common RGB input is shared for both targets. This choice improves both depth and detection estimations compared to single task models, as shown in the experiments. An approach similar to ours, applied to 3D bounding box detection, is presented in [28], where the authors train a three-loss model, sharing the feature extraction layers between the tasks.

In our system the obstacles bounding box regression part is obtained modifying the architecture of [29] making it fully convolutional. This allows for multiple bounding box predictions with a single forward pass. In addition, we also ask the obstacle detector to regress the average depth and the corresponding estimate variance of the detected obstacles.

Depth estimation is devised following the architecture of [9], improved by taking into account the obstacle detection branch. In particular, we correct the depth predictions by using the mean depth estimates computed by the obstacle detection branch to achieve robustness with respect to appearance changes. We prove the benefits of this strategy by validating the model in test sequences with different focal length and scene appearance. We compare our method to the ones of [11] and [12], showing a considerable increase of performances over these two baselines.

III. Network Overview

Our proposed network is depicted in Figure 2. Given an 256×160 RGB input, features are extracted with a fine-tuned version of the VGG19 network pruned of its fully connected layers [30]. VGG19 weights are initialized on the image classification task on the ImageNet dataset. Features are then fed to two, task-dependent branches: a depth prediction branch and a obstacle detector branch. The former is composed by 4 upconvolution layers and a final convolution layer which outputs the predicted depth at original input resolution. This branch, plus the VGG19 feature extractor, is equivalent to the fully convolutional network proposed in [9]. We optimize depth prediction on the following loss:

$$\begin{aligned} L_{depth} =& \frac{1}{n}\sum_i d_i^2 - \frac{1}{2n^2}(\sum_i d_i)^2 \\ &+ \frac{1}{n}\sum_i [\nabla_x D_i + \nabla_y D_i] \cdot N_i^* \end{aligned} \tag{1}$$

where $d_i = \log D_i - \log D_i^*$, D_i and D_i^* are respectively the predicted and ground truth depths at pixel i, N_i^* is the ground truth 3D surface normal, and $\nabla_x D_i$, $\nabla_y D_i$ are the horizontal and vertical predicted depth gradients. While the first two terms correspond to the scale invariant log RMSE loss introduced in [24], the third term enforces orthogonality between predicted gradients and ground truth normals, aiming at preserving geometrical coherence. With respect to the loss proposed in [8], that introduced a L2 penalty on gradients to the scale invariant loss, our loss performs comparably in preliminary tests.

The obstacle detection branch is composed by 9 convolutional layer with Glorot initialization. The detection methodology is similar to the one presented in [29]: the input image is divided into a 8×5 grid of square-shaped cells of size 32×32 pixels. For each cell, we train a detector to estimate:

- The (x, y) coordinates of the bounding box center
- The bounding box width w and height h
- A confidence score C
- The average distance of the detected obstacle from the camera m and the variance of its depth distribution v

The resulting output has a 40×7 shape. At test time, we consider only predictions with a confidence score over a certain threshold. We train the detector on the following loss:

$$\begin{aligned} L_{det} = \quad & \lambda_{coord}\sum_{i=0}^{N}[(x_i - x_i^*)^2 + (y_i - y_i^*)^2] \\ &+ \lambda_{coord}\sum_{i=0}^{N}[(w_i - w_i^*)^2 + (h_i - h_i^*)^2] \\ &+ \lambda_{obj}\sum_{i=0}^{N}(C_i - C_i^*)^2 + \lambda_{noobj}\sum_{i=0}^{N}(C_i - C_i^*)^2 \\ &+ \lambda_{mean}\sum_{i=0}^{N}(m_i - m_i^*)^2 + \lambda_{var}\sum_{i=0}^{N}(v_i - v_i^*)^2 \end{aligned} \tag{2}$$

where we set $\lambda_{coord} = 0.25$, $\lambda_{obj} = 5.0$, $\lambda_{noobj} = 0.05$, $\lambda_{mean} = 1.5$, $\lambda_{var} = 1.25$. Our network is trained simultaneously on both tasks. Gradients computed by each loss are backpropagated through their respective branches and the shared VGG19 multi-task feature extractor.

A. Exploiting detection to correct global scale estimations

The absolute scale of a depth estimation is not observable from a single image. However, learning-based depth estimators are able to give an accurate guess of the scale under certain conditions. While training, these models implicitly learn domain-specific object proportions and appearances. This helps the estimation process in giving depth maps with correct absolute scale.

As the relations between object proportions and global scale in the image strongly depend on camera focal length, at test time the absolute scale estimation are strongly biased towards the training set domain and its intrinsics. For these reasons, when object proportions and/or camera parameters change from training to test, scale estimates quickly degrade. Nonetheless, if object proportions stay roughly the same and only camera intrinsics are altered at test time, it is possible to employ some recovery strategy. If the size of a given object is known, we can analytically compute its distance from the camera and recover the global scale for the whole depth map. For this reason, we suppose that the obstacle detection branch can help recovering the global scale when intrinsics change. We hypothesize that, while learning to regress obstacles bounding boxes, a detector model implicitly learns sizes and proportions of objects belonging to the training domain. We can then evaluate estimated obstacle distances from the detection branch and use them as a tool to correct dense depth estimations. Let m_j be the average distance of the obstacle j computed by the detector, $\hat{D}_j$ the average depth estimation within the j-th obstacle bounding box, n_o the number of estimated obstacles, then we compute the correction factor k as:

$$k = \frac{\frac{1}{n_o}\sum_j^{n_o} m_j}{\frac{1}{n_o}\sum_j^{n_o} \hat{D}_j} \tag{3}$$

Finally, we calculate the corrected depth at each pixel i as $\tilde{D}_i = kD_i$. To validate our hypothesis, in Section IV-C we test on target domains with camera focal lengths that differ from the one used for training.

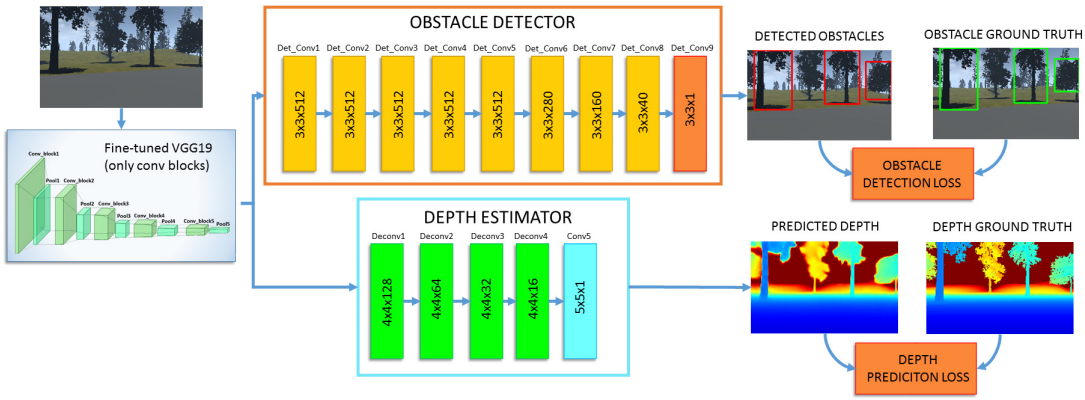


Fig. 2: Architecture of J-MOD^2. Given an RGB input, features are extracted by the VGG19 module and then fed into the depth estimation and obstacle detection branches to produce dense depth maps and osbtacles bounding boxes.

IV. Experiments

A. Datasets

1) UnrealDataset: UnrealDataset is a self-collected synthetic dataset that comprises of more than 100k images and 21 sequences collected in a bunch of highly photorealistic urban and forest scenarios with Unreal Engine and the AirSim plugin [31], which allows us to navigate a simulated MAV inside any Unreal scenarios. The plugin also allows us to collect MAV's frontal camera RGB images, ground truth depth up to 40 meters and segmentation labels. Some samples are shown in Figure 4(a). We postprocess segmentation labels to form a binary image depicting only two semantic classes: obstacle and non-obstacle by filtering these data with corresponding depth maps, we are finally able to segment obstacles at up to 20 meters from the camera and get ground truth labels for the detection network branch (Fig. 3). MAV's frontal camera has a horizontal field of view of 81,5 degrees.

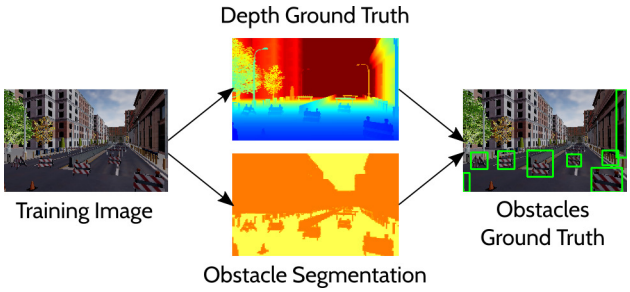


Fig. 3: Given depth and segmentation ground truth, we compute obstacle bounding boxes for each training image. We evaluate only obstacles in a 20 meters range.

2) Zurich Forest Dataset: Zurich Forest Dataset consist of 9846 real-world grayscale images collected with a hand-held stereo camera rig in a forest area. Ground truth depth maps are obtained for the whole dataset through semi-global stereo matching [32]. We manually draw 357 bounding boxes on a subset of 64 images to provide obstacle ground truth and evaluate detection in a real-world scenario.

B. Training and testing details

As baselines, we compare J-MOD^2 with:

- The depth estimation method proposed in [9].
- Our implementation of the multi-scale Eigen's model [8].
- A simple obstacle detector, consisting of our proposed model, trained without the depth estimation branch.
- Our implementation of the multi-modal autoencoder (later referred as Full-MAE) proposed by Cadena et al. [11].
- Our implementation of the joint refinement network (later referred as JRN) proposed by Jafari et al. [12].

We train J-MOD^2 and all the baseline models on 19 sequences of the UnrealDataset. We left out sequences 09 and 14 for testing. All the approaches have been trained on a single NVIDIA Titan X GPU. Training is performed with Adam optimizer by setting a learning rate of 0.0001 until convergence. The segmentation tasks for the Full-MAE and the JRN baselines are trained to classify two classes: "obstacle" and "not obstacle". The JRN is trained to fuse and refine depth estimations from our implementation of [8] with segmentation estimates from the SotA segmentation algorithm of Long et al. [33], as suggested by the authors, with the latter retrained on the 2-class segmentation problem of the UnrealDataset.

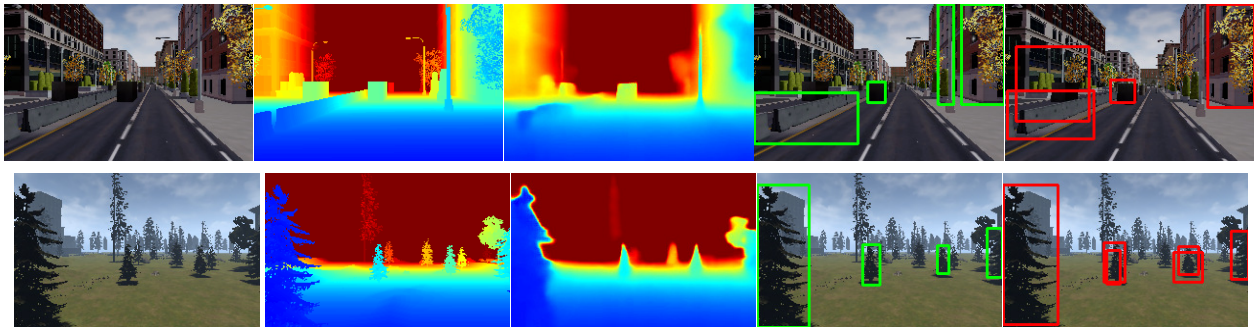
At test time, all baseline methods are tested using only RGB inputs. For both methods, we then infer obstacle bounding boxes from their depth and segmentation estimates applying the same procedure described in Figure 3, allowing direct comparison with our method. All the approaches are tested on the test sequences of the UnrealDataset and on the whole Zurich Forest Dataset. Note that, while testing on the latter, we do not perform any finetuning for both our method and the baselines.

At runtime, estimations require about 0.01 seconds per frame on a NVIDIA Titan X GPU. We also test J-MOD^2 on a NVIDIA TX1 board, to evaluate its portability on a on-board embedded system, measuring an average forward time of about 0.28 seconds per frame. The code for J-MOD^2 and all the baseline methods is available online[1]

To evaluate the depth estimator branch performance, we compute the following metrics:

- Linear RMSE and Scale Invariant Log RMSE ($\frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}(\sum_i d_i)^2$, with $d_i = \log y_i - \log y_i^*$) on the full depth map.

[1]http://isar.unipg.it/index.php?option=com_content&view=article&id=47&catid=2&Itemid=188



(a) RGB Input (b) Depth GT (c) Depth Estimation (d) Obstacle GT (e) Detected Obstacles

Fig. 4: J-MOD^2 qualitative results on the UnrealDataset.

	DEPTH [9]	DETECTOR	EIGEN [8]	FULL-MAE [11]	JRN [12]	J-MOD^2	
RMSE Full Depth Map	3.653	-	3.785	7.566	7.242	**3.473**	Lower
Sc.Inv RMSE Full Depth Map	0.042	-	0.043	0.124	0.110	**0.036**	is
Depth RMSE on Obs.(Mean/Var)	1.317 / 37.124	-	1.854 / 50.71	5.355/180.67	2.938 / 87.595	**1.034 / 29.583**	better
Detection RMSE on Obs.(Mean/Var)	-	2.307/ 59.407	-	-	-	**1.754 / 46.006**	
Detection IOU	-	63.11%	-	32.58%	44.19%	**66.58%**	Higher
Detection Precision	-	72.15%	-	75.53%	54.37%	**78.64%**	is
Detection Recall	-	90.05%	-	44.38%	49.55%	**90.85%**	better

TABLE I: Results on the UnrealDataset. For the depth estimation task we report full depth map RMSE and scale invariant errors, obstacle-wise depth and detection branches statistics (mean/variance) estimation errors and detector's IOU, precision and recall.

- Depth RMSE on Obstacles (Mean/Variance): For each ground truth obstacle, we compute its depth statistics (mean and variance) and we compare them against the estimated ones by using linear RMSE.

For the detector branch, we compute the following metrics:

- Detection RMSE on Obstacles (Mean/Variance):For each detected obstacle, we compare its estimated obstacle depth statistics (mean and variance) with the closest obstacle ones by using linear RMSE.
- Intersection Over Union (IOU)
- Precision and Recall.

C. Test on UnrealDataset

We report results on Table I. For [9] and [8] we report results only on depth-related metrics, as they do not perform any detection. Results confirm how J-MOD^2 outperforms all the other baselines in all metrics, corroborating our starting claim: object structures learned by the detector branch improve obstacles depth estimations of the depth branch. At the same time, localization and accuracy of the detected bounding boxes improve significantly compared to our single-task obstacle detector. We achieve good performances on both urban and forest sequences, without any significant discrepancy due to different depicted objects and contexts. We report qualitative results on Figure 4. According to the results on the NYU benchmark reported in [12], we expect JRN to outperform [8] on depth metrics, but this is not observed in this experiment. Our intuition is that the JRN segmentation network deals with a more challenging scenario, since the labels to the different objects are simply "obstacle", "not-obstacle" while in the original NYU there were specific labels for each object category. This makes this task for JRN similar to a semi-supervised learning problem, that is implicitly more difficult. Our system relies on a obstacle detector, that is a much simpler task to train, and therefore has an edge in this scenario.

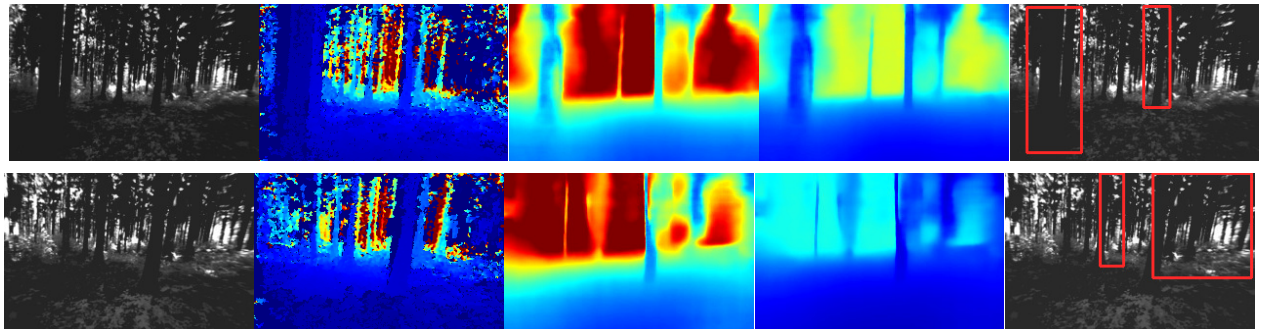
To validate our proposed depth correction strategy introduced in Section III-A, we also simulate focal length alterations by cropping and upsampling a central region of the input images of the UnrealDataset. We evaluate performances on different sized crops of images on the sequence-20, one of the training sequences, comprising of more than 7700 images. We choose to stage this experiment on a training sequence to minimize appearance-induced error and make evident the focal-length-induced error. We report results on Table II. When no crop is applied, camera intrinsics are unaltered and appearance-induced error is very low, as expected. As correction is applied linearly on the whole depth map, when scale-dependant error is absent or low, such correction worsen estimations by 19% on non-cropped images. A 230×144 crop simulates a slightly longer focal length. All metrics worsen, as expected, and correction still cause a 15% higher RMSE error. When 204×128 crops are evaluated, correction starts to be effective, improving performances by $1,45\%$ with respect to the non-corrected estimation. On 154×96 crops, correction leads to a 23% improvement. On 128×80 crops, correction improves performance by 25%. We also observe how the detection branch outperforms the depth estimation branch on obstacle distance evaluation as we apply wider crops to the input. This results uphold our hypothesis that detection branch is more robust to large mismatches between training and test camera focal lengths and can be used to partially compensate the induced absolute scale estimation deterioration.

D. Test: Zurich Forest Dataset

In this experiment we test our models, trained on synthetically generated data, on a real world scenario without performing any finetuning, to verify the generalization capabilities of the models when tested on unseen domains. Depth metrics (Linear RMSE and Scale Invariant MSE) refer to the whole dataset, while all the other metrics refer to the labelled subset, as described in Section IV-A2. Results are reported

	ORIGINAL SIZE		CROP 230X144		CROP 204X128		CROP 154X96		CROP 128X80	
	NoCor	Cor	NoCor	Cor	NoCor	Cor	NoCor	Cor	NoCor	Cor
RMSE Full Depth Map	**2.179**	2.595	**2.632**	3.042	4.052	**3.991**	8.098	**6.234**	10.825	**8.045**
Sc. Inv RMSE Full Depth Map	**0.096**	0.115	**0.121**	0.134	0.173	**0.164**	0.274	**0.217**	0.305	**0.250**
Depth RMSE on Obs.(Mean)	**0.185**	0.676	**1.293**	1.458	2.465	**2.219**	4.865	**3.583**	6.148	**4.485**
Detector RMSE on Obs.(Mean)	0.404		1.079		1.998		4.124		5.450	

TABLE II: Results of J-MOD^2 on the sequence-20 of the UnrealDataset on different-sized central crops. For each crop, we report in bold the better estimation between unchanged (labeled as NoCor) and corrected depths (labeled as WithCor).



(a) RGB Input (b) Depth GT (c) Non Corrected Depth Estimation (d) Corrected Depth Estimation (e) Detected Obstacles

Fig. 5: J-MOD^2 qualitative results on the Zurich Forest Dataset.

	DEPTH [9]		DETECTOR		EIGEN [8]		FULL-MAE [11]		JRN [12]		J-MOD^2	
	Cor	NoCor	Cor	NoCor	Cor	NoCor	Cor	NoCor	Cor	NoCor	Cor	NoCor
RMSE	-	12.421	-	-	-	14.640	-	17.581	-	10.114	**9.009**	12.569
Sc. Inv RMSE	-	0.873	-	-	-	1.025	-	1.711	-	0.702	**0.429**	0.954
Depth RMSE on Obs.(Mean)*	-	**4.378**	-	-	-	8.060	-	10.488	-	4.783	4.510	4.847
Detector RMSE on Obs.(Mean)*	-		6.277		-		-		-		**3.702**	
Detector IOU*	-		14.4%		-		2.13%		9.19%		**26.32%**	
Detector Precision*	-		25.32%		-		11.4%		13.18%		**48.36%**	
Detector Recall*	-		10.80%		-		1.12%		6.72%		**20.49%**	

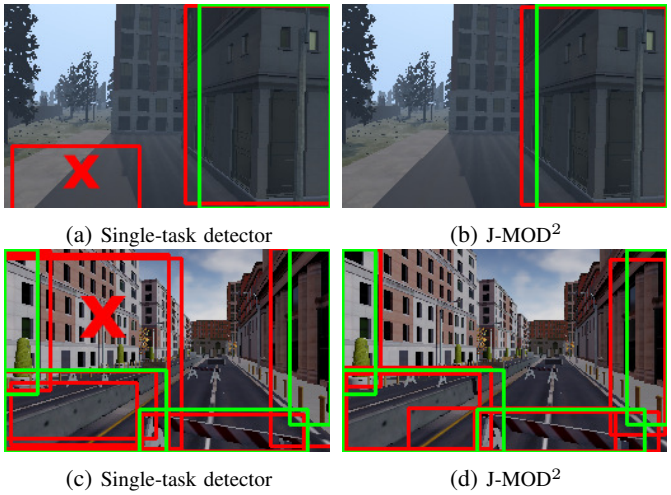
TABLE III: Results on the Zurich Forest Dataset. Metrics marked with a * symbol are evaluated on a subset of 64 images with ground truth bounding boxes.

on Table III. J-MOD^2 outperforms all baselines in almost all metrics, which suggests improved generalization capabilities. Furthermore, we show how the correction factor introduced in Section III-A improves J-MOD^2 depth estimation by about 28% on the RMSE metric, reducing the scale-induced errors on the estimates caused by the different camera parameters. We report qualitative results on Figure 5. The performance of all the approaches are lower with respect to the UnrealDataset. This is expected, since the synthetic textures and general appearance are different from the ones in this dataset. In addition, the camera characteristics do not match the ones of the UnrealDataset sequences.

E. Qualitative analysis of the multi-task interaction

Besides the advantages given by J-MOD^2 in terms of numerical performance, in the following, we qualitatively discuss the benefits of our joint architecture compared to its single task counterparts.

Figure 6 shows a comparison between the estimated obstacle bounding boxes of the detector-only architecture and the J-MOD^2 ones. It can be observed that, by exploiting the auxiliary depth estimation task, J-MOD^2 learns a detector that is aware of scene geometry. This results in an architecture that models a better concept of obstacle and, thus, is more precise in detecting what really determines a threat for the robot. Hence, it avoids wrong detections, such as ground surfaces

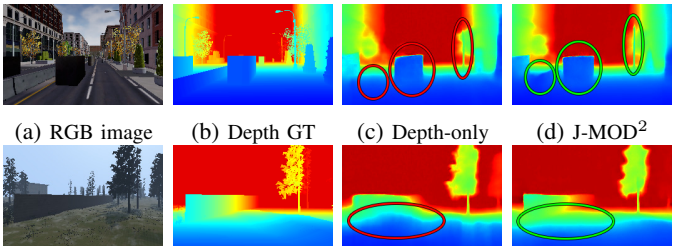


(a) Single-task detector (b) J-MOD^2

(c) Single-task detector (d) J-MOD^2

Fig. 6: For each row, we compare J-MOD^2 obstacle detections with the detector-only architecture. Ground truth bounding boxes are reported in green, predictions in red. In the first example (first row), the single-task detector erroneously detects a false obstacle on the ground. Similarly, in the second example (second row), the single-task wrongly considers the whole building on the left as an obstacle while only its closest part is an immediate threat for robot navigation.

(see Figures 6(a) and 6(b)), or full buildings of which only the closest part would constitutes an immediate danger for navigation (see Figures 6(c) and 6(d)).



(a) RGB image (b) Depth GT (c) Depth-only (d) J-MOD^2

(e) RGB image (f) Depth GT (g) Depth-only (h) J-MOD^2

Fig. 7: For each row, we compare J-MOD^2 depth maps with the ones predicted by the depth-only architecture. J-MOD^2 estimations are sharper and more defined. Consider, for example, the bollard and the lamppost in Figures 7(a)-7(d) or the ground surface in Figures 7(e)-7(h), whose depth is wrongly estimate by the depth-only estimator.

Similarly, depth estimation branch of the proposed J-MOD^2 approach takes advantage from the obstacle detector task to refine the estimation of the scene geometry. The representation learned by the J-MOD^2 depth estimation stream contains also visual clues about object shapes and proportions, which gives it the capability to integrate object semantics when estimating the scene depths. Compared to the depth-only architecture [9], our approach predicts sharper and more precise depth maps. This is more evident if we consider very thin elements and objects that could be mistaken for ground surfaces (*e.g.* consider the lamppost and the bollard in Figures 7(a)7(d) or the ground estimates in Figures 7(e)7(h)).

F. Navigation experiments

We further validate J-MOD^2 effectivness for obstacle detection applications by setting up a simulated full MAV navigation system. We depict the system architecture in Figure 8. We create a virtual forest scenario on Unreal Engine, slightly different from the one used for dataset collection. The line-of-sight distance between the takeoff point and the designed landing goal is about 61 meters. Trees are about 6 meters tall and spaced 7 meters from each other, on average. An aerial picture of the test scenario is reported in Figure 8.

A simulated MAV is able to navigate into the scenario and collect RGB images from its frontal camera. We estimate depth from the captured input and we employ it to dynamically build and update an Octomap [34]. We plan obstacle-free trajectories exploiting an off-the shelf implementation of the RRT-Connect planner [35] from the *MoveIt!* ROS library, which we use to pilot the simulated MAV at a cruise speed of $1m/s$. Trajectories are bounded to a maximum altitude of 5 meters. As a new obstacle is detected along the planned trajectory, the MAV stops and a new trajectory is computed. The goal point is set 4 meters above the ground. For each flight, we verify its success and measure the flight distance and duration. A flight fails if the MAV crashes or gets stuck, namely not completing its mission in a 5 minute interval. We compare J-MOD^2 with the Eigen's baseline, both trained on the UnrealDataset.

While planning, we add a safety padding on each Octomap obstacles. This enforces the planner to compute trajectories not too close to the detected obstacles. For each estimator, we set this value equal the average RMSE obstacle depth error on the UnrealDataset test set, as reported in Table I: 1.034 meters for J-MOD^2, 1.854 meters for Eigen. We refer to this value as a reliability measure of each estimator; the less accurate an estimator is, the more padding we need to guarantee safe operation. We perform 15 flights for each depth estimator and report their results on Table IV.

	EIGEN [8]	J-MOD^2
Success rate	26,6%	**73,3%**
Failure cases	8 stuck / 3 crash	2 stuck / 2 crash
Avg. flight time	147s	**131s**
Std. Dev. Flight Time	18.51s	**12.88s**
Avg. flight distance	78m	**77m**
Std. Dev. Flight Distance	**4.47m**	9.95m

TABLE IV: Results of the navigation experiment. We compare the navigation success rate when using J-MOD^2 and Eigen's approach as obstacle detection systems.

J-MOD^2 clearly performs better in all metrics, proving that how our method is effective for monocular obstacle detection. By analyzing failure cases, for 6 times the MAV using Eigen as obstacle detector got stuck in the proximity of goal point because ground was estimated closer than its real distance, causing planner failure in finding an obstacle-free trajectory to the goal. J-MOD^2 failures are mostly related on erratic trajectory computation which caused the MAV to fly too close to obstacles, causing lateral collisions or getting stuck in proximity of tree's leaves.

V. Conclusion and Future Work

In this work, we proposed J-MOD^2, a novel end-to-end deep architecture for joint obstacle detection and depth estimation. We demonstrated its effectiveness in detecting obstacles on synthetic and real-world datasets. We tested its robustness to appearance and camera focal length changes. Furthermore, we deployed J-MOD^2 as an obstacle detector and 3D mapping module in a full MAV navigation system and we tested it on a highly photo-realistic simulated forest scenario. We showed how J-MOD^2 dramatically improves mapping quality in a previously unknown scenario, leading to a substantial lower navigation failure rate than other SotA depth estimators. In future works, we plan to further improve robustness over appearance changes, as this is the major challenge for the effective deployment of these algorithms in practical real-world scenarios.

References

[1] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[2] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4557–4564.

[3] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.

[4] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion- and uncertainty-aware path planning for micro aerial vehicles," *Journal of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

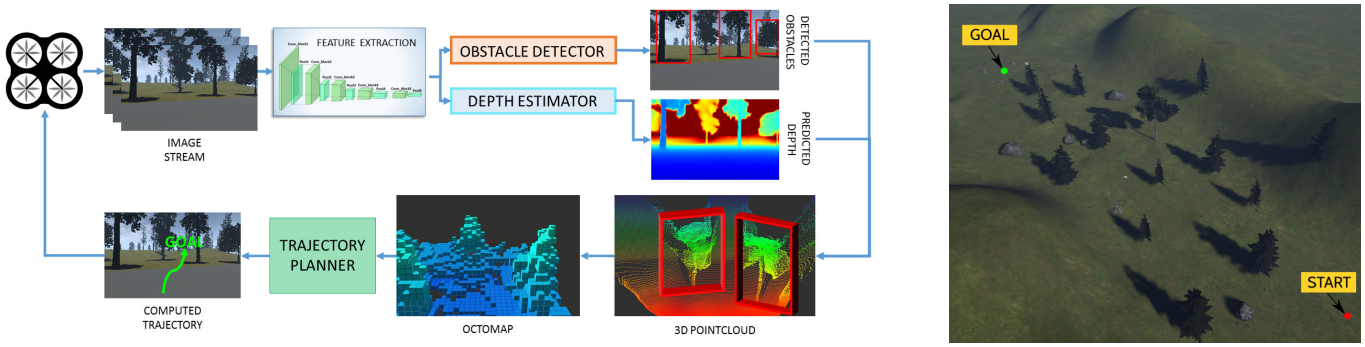


Fig. 8: Architecture of the full navigation pipeline (on the left) and a aerial picture of the test scenario (on the right). For each RGB image captured by the MAV frontal camera, a depth map is computed and converted into a point cloud used to update the 3D map and compute an obstacle-free trajectory. The MAV then flies along the computed trajectory until a new obstacle is detected.

[5] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, *et al.*, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[6] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[8] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[9] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Towards domain independence for learning-based monocular depth estimation," *IEEE Robotics and Automation Letters*, 2017.

[10] S. Yang, S. Konam, C. Ma, S. Rosenthal, M. Veloso, and S. Scherer, "Obstacle avoidance through deep networks based intermediate perception," *arXiv preprint arXiv:1704.08759*, 2017.

[11] C. Cadena, A. Dick, and I. Reid, "Multi-modal auto-encoders as joint estimators for robotics scene understanding," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[12] O. H. Jafari, O. Groth, A. Kirillov, M. Y. Yang, and C. Rother, "Analyzing modular CNN architectures for joint depth prediction and semantic segmentation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4620–4627.

[13] C. Nous, R. Meertens, C. De Wagter, and G. de Croon, "Performance evaluation in obstacle avoidance," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 3614–3619.

[14] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3046–3052.

[15] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2609–2616.

[16] H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers, "Collision avoidance for quadrotors with a monocular camera," in *Experimental Robotics*. Springer, 2016, pp. 195–209.

[17] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.

[18] D. P. Frost, O. Khler, and D. W. Murray, "Object-aware bundle adjustment for correcting monocular scale drift," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4770–4776.

[19] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[20] S. Daftry, S. Zeng, A. Khan, D. Dey, N. Melik-Barkhudarov, J. A. Bagnell, and M. Hebert, "Robust monocular flight in cluttered outdoor environments," *CoRR*, vol. abs/1604.04779, 2016.

[21] A. Beyeler, J.-C. Zufferey, and D. Floreano, "Vision-based control of near-obstacle flight," *Autonomous robots*, vol. 27, no. 3, pp. 201–219, 2009.

[22] C. Bills, J. Chen, and A. Saxena, "Autonomous mav flight in indoor environments using single image perspective cues," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 5776–5783.

[23] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1750–1757.

[24] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[25] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, Oct 2016.

[26] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, "Cnn-based single image obstacle avoidance on a quadrotor," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 6369–6374.

[27] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *CoRR*, vol. abs/1705.07115, 2017.

[28] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5632–5640.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[31] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *arXiv preprint arXiv:1705.05065*, 2017.

[32] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[33] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

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

[35] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.