Fast Depth Prediction and Obstacle Avoidance on a Monocular Drone Using Probabilistic Convolutional Neural Network

Xin Yang, *Member, IEEE*, Jingyu Chen, Yuanjie Dang, Hongcheng Luo, Yuesheng Tang, Chunyuan Liao, Peng Chen, and Kwang-Ting Cheng, *Fellow, IEEE*

***Abstract*— Recent studies employ advanced deep convolutional neural networks (CNNs) for monocular depth perception, which can hardly run efficiently on small drones that rely on low/middle-grade GPU(e.g. TX2 and 1050Ti) for computation. In addition, the methods which can effectively and efficiently produce probabilistic depth prediction with a measure of model confidence have not been well studied. The lack of such a method could yield erroneous, sometimes fatal, decisions in drone applications (e.g. selecting a waypoint in a region with a large depth yet a low estimation confidence). This paper presents a real-time onboard approach for monocular depth prediction and obstacle avoidance with a lightweight probabilistic CNN (pCNN), which will be ideal for use in a lightweight energy-efficient drone. For each video frame, our pCNN can efficiently predict its depth map and the corresponding confidence. The accuracy of our lightweight pCNN is greatly boosted by integrating sparse depth estimation from a visual odometry into the network for guiding dense depth and confidence inference. The estimated depth map is transformed into Ego Dynamic Space (EDS) by embedding both dynamic motion constraints of a drone and the confidence values into the spatial depth map. Traversable waypoints are automatically computed in EDS based on which appropriate control inputs for the drone are produced. Extensive experimental results on public datasets demonstrate that our depth prediction method runs at 12Hz and 45Hz on TX2 and 1050Ti GPU respectively, which is 1.8X∼5.6X faster than the state-of-the-art methods and achieves better depth estimation accuracy. We also conducted experiments of obstacle avoidance in both simulated and real environments to demonstrate the superiority of our method to the baseline methods.**

***Index Terms*— Depth prediction, convolutional neural network, adversarial learning, obstacle avoidance, drone platform.**

I. Introduction

SMALL drones have the advantages of greater flexibility, lower power consumption, and cheaper prices than the large ones, and are becoming increasingly popular for a wide range of applications, including package delivery, filming and surveillance in cramped and cluttered environments. The advantages of small drones, together with the considerable advances in integrated circuit design and embedded systems, stimulate the rapid development of miniaturizing drones and increasing their processing capability.

Perceiving depth and avoiding obstacles are essential tasks for intelligent navigation of small drones. Conventional solutions typically employ distance sensors such as ultrasound sensors, radar, Microsoft Kinect or stereo cameras. However, such sensors could either incur a high cost and power consumption or have a limited working range and environment, making them unsuitable for small drones. Using a monocular camera as the only depth sensor for perceiving and avoiding obstacles becomes a very attractive solution for miniature drones due to several advantages of the monocular camera, including simplicity to use, small footprint, light weight, low cost, and low power consumption.

Several systems have successfully demonstrated the feasibility of using a frontal monocular camera to estimate depth and avoid obstacles on a small drone. For instance, in [1] and [2], the authors divided an image into patches, represented each patch using a set of handcrafted features and estimated the depth of each patch using a pre-trained SVM classifier. Based on the estimated depth, waypoints/control commands are generated based on heuristic schemes [3] or reinforcement learning [2]. In [4], the authors formulated a Markov Random Field (MRF) classification model for estimating obstacles. However, existing systems can only achieve a relatively low onboard speed for depth inference, e.g. 5Hz in [1], which limits the reactive speed of a drone and in turn limits its use in some emergent applications. In addition, the accuracy of single image depth estimation based on handcrafted features is usually poor, limiting the application of existing systems in highly cramped and cluttered environments.

To achieve satisfactory depth estimation accuracy, advanced deep CNN for end-to-end feature extraction and

Manuscript received March 3, 2019; revised September 9, 2019; accepted November 18, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61872417, in part by the Wuhan Science and Technology Bureau under Award 2017010201010111, in part by the Hubei Provicinal Natural Science Foundation under Grant ZRMS2017000375, in part by the Fundamental Research Funds for the Central Universities under Grant 2019kfyRCPY118, and in part by the Program for HUST Acadamic Frontier Youth Team. The Associate Editor for this article was H. G. Jung. *(Corresponding author: Peng Chen.)*

X. Yang, J. Chen, and H. Luo are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China.

Y. Dang and Y. Tang are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310058, China.

C. Liao is with HiScene Information Technology Company, Ltd., Shanghai 200120, China.

P. Chen is with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310058, China (e-mail: chenpeng@zjut.edu.cn).

K.-T. (Tim) Cheng is with the School of Engineering, The Hong Kong University of Science and Technology, Hong Kong.

Digital Object Identifier 10.1109/TITS.2019.2955598

depth regression have been explored recently. For instance, Eigen et al. [5], [6] proposed a two-stage CNN where the first stage predicts a global coarse depth image, followed by refinement to the prediction locally in the second stage. In [7]. Laina et al. developed a fully convolutional network architecture with residual learning to model *intensity-to-depth* mapping. Their network considerably boosted the accuracy of single image depth estimation on several benchmark datasets. Along the direction of using CNN, several other works [8]–[10] have been developed to leverage unsupervised or semi-supervised learning to alleviate the requirement of labeled depth images in supervised learning of CNN models. To improve the depth estimation accuracy, the authors in [11], [12] integrated Conditional Random Field (CRF) into the network. Several studies [13]–[15] proposed to integrate monocular visual SLAM with single image depth estimation. For instance, in [15] Luo et al. collected training data on-the-fly based on the camera pose estimated from visual SLAM and updated the CNN model progressively for unseen scenes. Tateno *et al.* [13] fused CNN-predicted dense depth maps with semi-dense depth maps obtained from direct monocular SLAM. Similarly, in [14], the authors designed a CNN which takes RGB images and semi-dense depth maps from SLAM and outputs a dense depth map.

Existing CNN-based methods [5]–[7], [11]–[15] are primarily designed for achieving a high depth estimate accuracy. As a result, these methods typically employ very deep network architectures and in turn yield a high computational cost and great difficulties in running efficiently on a low/middle grade GPU available in small drones. In addition, the measurement of model confidence is critical for depth prediction systems, based on which we can tell how reliable a depth prediction output can be counted on for decision making. For instance, mistakenly selecting a waypoint in a region with a large depth estimate yet a low confidence could lead to drone collision. Several methods [1] address this issue by subtracting an average estimation error from depth inference and select waypoints in the "*confined*" depth map. However, the depth estimation error could vary significantly for different scenes, viewpoints or even different regions in an image. Confining candidate waypoint selection regions using an averaging error could either exclude too many collision-free regions for waypoint selection or incorrectly select waypoints that lead to collision. Several methods also proposed to estimate the confidence map of depth prediction. Kendall and Gal [16] presented a Bayesian deep learning framework to combine two major types of uncertainty, i.e. aleatoric uncertainty and epistemic uncertainty. Tian and Li [17] proposed two parallel networks to separately predict depth map and confidence map. The confidence map was trained with a patch-based cost function from the depth network as supervision. However, the methods which can effectively and efficiently produce probabilistic depth prediction with a measure of model confidence have not been well studied. It is worth explaining that our pCNN is different from [18], in which a probability map is generated to indicate the importance of the weights of a CNN. Unimportant weights were pruned to accelerate computation. In comparison, our pCNN generates a probability map to denote the confidence of a depth estimate being correct. Moreover, existing methods [13]–[15] utilized semi-dense depth maps from direct SLAM, e.g. LSD-SLAM, for boosting accuracy of dense depth estimation. ORB-SLAM [19] has been widely acknowledged as more accurate, robust and efficient than LSD-SLAM. However, ORB-SLAM can provide only sparse depth maps. How to efficiently combine ORB-SLAM with CNN and utilize sparse depth measurements from ORB-SLAM for guiding accurate dense depth prediction has not been studied.

In this paper, we for the first time demonstrate an energy-efficient small drone system equipped with a TX2 GPU and Intel NUC core i5 CPU that can run at 12Hz onboard for depth prediction and obstacle avoidance with the state-of-the-art accuracy. A key contribution of our system lies in a novel pCNN, which employs a lightweight and shallow network architecture to efficiently predict a depth map for each video frame and meanwhile outputs the corresponding confidence for each depth estimate. The accuracy of our pCNN is greatly boosted by taking advantage of sparse depth estimation from a visual odometry running in parallel with CNN depth prediction. Specifically, our pCNN uses a sparse depth as guidance to supervise dense depth prediction of a single RGB image. The network is trained based on a combination of three loss functions, i.e. an adversarial loss for ensuring a high similarity of geometric structure between predicted depth and true depth, a pixel-wise mean square error (MSE) loss for achieving high accuracy of average absolute depth values and a pixel-wise L1 loss for ensuring reliable confidence estimation. Majority features for depth and confidence estimation are sharing in the network for efficiency in inference. The estimated confidence values are embedded into the depth map, together with the dynamic motion constraints of a drone, yielding an Ego Dynamic Space (EDS) based on which traversable waypoints are automatically computed and appropriate control inputs for the drone are produced. We built a highly energy-efficient drone using an Intel NUC as the main computing platform which is further integrated with an NVIDIA Jetson TX2 graphics card. Extensive experimental results on public datasets demonstrate that our depth prediction method runs at 12Hz on our drone which is 1.8X$\sim$5.6X faster than the state-of-the-art methods [7], [13], [14] and achieves greater depth estimation accuracy. Experimental results in both simulated and real environments demonstrate the effectiveness and efficiency of our drone system for automated obstacle avoidance.

The rest of the paper is organized as follows. Sec. II presents our lightweight monocular depth prediction algorithm, followed by the obstacle avoidance method in Sec. III. In Sec. IV we describe the entire platform design, including the hardware structure, PX4 autopilot, robot operating system (ROS), and their connection micro air vehicle communication protocol on ROS (MAVROS). Sec. V discusses the experimental results and Sec. VI concludes the paper.

II. Monocular Depth Prediction

Fig. 1 illustrates the framework of our monocular depth and confidence prediction using the probabilistic CNN (pCNN).

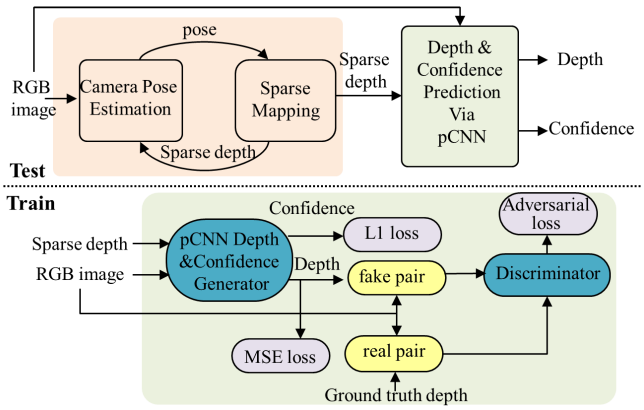


Fig. 1. Framework of our monocular video-based depth and confidence prediction based on pCNN.

Our method mainly consists of two modules: 1) sparse depth map generation via feature tracking and matching given a sequence of RGB images from a monocular camera; 2) depth and confidence prediction via the pCNN generator based on a RGB image and the corresponding sparse depth map as inputs. The pCNN is trained based on an adversarial loss and a pixel-wise mean squared error (MSE) loss for depth estimation and an L1 based loss for confidence prediction. In the following, we describe each module in detail.

A. Sparse Depth Map Generation

Various methods can be utilized for sparse depth map generation. In this work, we leveraged the feature tracking based approach used in ORB-SLAM [19] due to its low computational complexity, high accuracy and robustness. Similar as ORB-SLAM, we initialize the camera pose (i.e. $\boldsymbol{T}_w^i \in SE(3)$) via the obtained fundamental matrix or homography matrix. Specifically, given pairs of ORB matching points between two images, we calculate the fundamental matrix $\boldsymbol{F}$ via the 8-point algorithm [20] and then solve the camera pose $\boldsymbol{T}$ via SVD decomposition of $\boldsymbol{F}$. Meanwhile, we also calculate the homography matrix via the RANSAC algorithm [21] based on the given pairs of ORB matching points. To initialize the camera pose, we follow the strategy in ORB-SLAM to compute a R_H score for evaluating the likelihood of a current scene to be planar. If $R_H > 0.45$, we choose the homography matrix which adequately captures the planar and low parallax cases. Otherwise, we select the fundamental matrix. Map points are then initialized based on the estimated poses and triangulation. Specifically, we formulate two reprojection equations as an equation set,

$$\boldsymbol{KP} = \mathbf{p}_1, \boldsymbol{KTP} = \mathbf{p}_2 \tag{1}$$

where $\boldsymbol{K}$ is the intrinsic matrix of the camera, $\mathbf{p}_1$ and $\mathbf{p}_2$ are matching ORB points on image plane. The coordinate of $\boldsymbol{P} = [X, Y, Z]^T$ can be obtained by solving this equation set and Z corresponds to the depth of point $\boldsymbol{P}$.

Once the initialization step succeeds, the camera poses of the following frames are estimated by computing correspondences with ORB features associated to map points in each frame I_i and then performing the RANSAC and PnP algorithms [22] to find a camera pose. If sufficient inliers can be found for an estimated camera pose, the estimated pose is applied to guide a search for more correspondences with the map points and then the camera pose is optimized again based on the newly founded matches. With the calculated camera pose, new map points from current frames are created by triangulating the feature points in the covisibility map. Then the camera poses and map points are jointly optimized with local bundle adjustment. Once the tracking of a new frame is finished, we reproject the map points within its corresponding keyframe to the current frame based on its pose $\boldsymbol{T}$, thus generate the sparse depth map of the current frame.

B. pCNN for Depth and Confidence Estimation

Fig. 2 shows the architecture of our pCNN for a depth and confidence predictor $\boldsymbol{G}$. Specifically, our prediction network first employs two separate input branches to take both an RGB image I and its corresponding sparse depth map D_s as inputs. Each branch processes the respective input using a convolutional layer, followed by a max pooling layer. The outputs of the two branches are concatenated together and processed by another convolutional layer. After that, 8 successive residual modules [23] of ResNet18 are applied to perform further feature extraction. The obtained feature maps are then resized by a set of de-convolutional layers. In order to capture the fine details of an image, skip connections are applied in our network to form a 'U-NET' structure. Finally, the feature maps are processed by two separate branches, each of which consists of a convolutional layer to generate an individual output (i.e. a depth map and a confidence map).

The design goal of our pCNN is to intelligently combine information of an RGB image and a sparse map so that the two inputs mutually benefit each other for accurate depth and confidence prediction. The input sparse depths provide a rough geometric structure of a scene to guide the process of learning correlations among pixel intensities from an RGB image, depth values and the model confidence; the learned *intensity-depth-confidence* knowledge in turn adjusts depths of sparse pixels according to their intensities. The sophisticated relationships among sparse depths, pixel intensities, true depths and confidence are captured in the learning process of minimizing our loss function.

To this end, in this work we train our prediction network $\boldsymbol{G}$ as a feed-forward CNN $\boldsymbol{G}_{\boldsymbol{\theta}_G}(I, D_s)$ parameterized by $\boldsymbol{\theta}_G$. Here $\boldsymbol{\theta}_G = \{W_{1:L}; b_{1:L}\}$ denotes the weights and biases of a deep network with L layers. For training images $I_n^{Train}, n = 1, \ldots, N$ with the corresponding sparse depth maps $D_{s,n}^{Train}$ and dense depth maps $D_{d,n}^{Train}$, we solve:

$$\widehat{\boldsymbol{\theta}}_G = \text{argmin} \frac{1}{N} \sum_{n=1}^{N} l(\boldsymbol{G}_{\boldsymbol{\theta}_G}(I_n^{Train}, D_{s,n}^{Train}), D_{d,n}^{Train}) \tag{2}$$

As mentioned above, the definition of the loss function l for training $\boldsymbol{G}_{\boldsymbol{\theta}_G}(I, D_s)$ is critical for the accuracy of the output depth and confidence. In this work, we formulate the loss l as the weighted summation of a confidence loss l_C and a depth

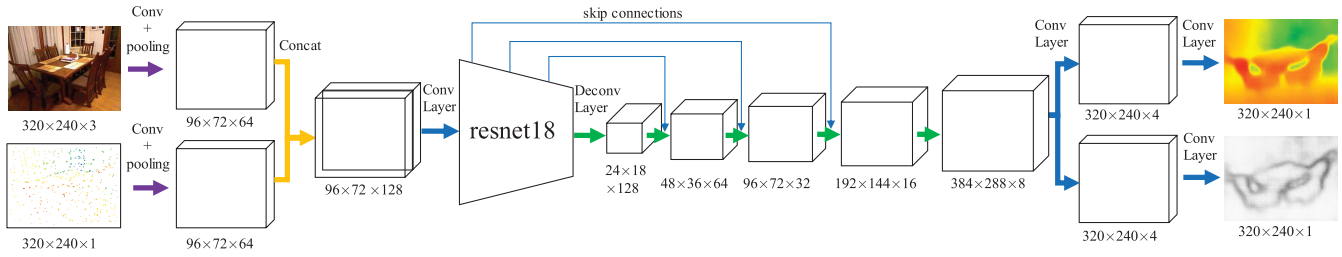


Fig. 2. Network architecture of our pCNN.

loss l_D(which consists of two parts, i.e. pixel-wise MSE loss l_D^{MSE} and adversarial loss l_D^{Adv}) as:

$$l = \alpha l_D + \beta l_C = \alpha(\delta l_D^{MSE} + \gamma l_D^{Adv}) + \beta l_C \tag{3}$$

In the following we describe the design of the losses in detail.

1) Depth Loss l_D: The pixel-wise MSE loss is a common choice for single image depth estimation based on supervised learning. However, minimizing the pixel-wise MSE loss encourages finding pixel-wise averages of plausible solutions which often yield an overly smoothed depth map that lacks high-frequency content. Therefore, in addition to the MSE loss, we also add an adversarial loss l_D^{Adv} which is the output from a discriminator.

Our objective is that minimizing l_D^{Adv} with respect to $\boldsymbol{\theta_G}$ would encourage our generator to favor solutions that reside on the manifold of true depth maps and meanwhile well capture the *intensity-to-depth* correlations between images and the corresponding true depth maps. To this end, we designed our discriminator $\boldsymbol{D}$ to approximate the distance between the distribution of *image-true depth* pairs (i.e. real pairs) and that of *image-generated depth* pairs (i.e. fake pairs). Specifically, the network of our discriminator takes an image pair consisting of an RGB image and a depth map as input and concatenates the input to form four-channel feature maps for further processing. The discriminator contains 7 convolutional layers with 4 × 4 filter kernels. Each of the first 4 convolutional layers reduces the size of features maps by a stride of 2 pixels and concurrently increases the number of feature maps. The resulting 512 feature maps are followed by 3 convolutional layers with a stride of 1 pixel, resulting in 32 feature maps. After that, two fully connected layers are applied to generate a single value as the output. We followed the architectural guidelines summarized by Radford *et al.* [24] to replace all the RELU activations by the Leaky RELU activations and avoid using pooling layers and batch-normalization layers throughout the network.

We calculate the recently proposed Wasserstein distance (W-distance) [25] between the fake and real pairs by maximizing $W(\boldsymbol{\theta_G}, \boldsymbol{\theta_D})$ w.r.t. $\boldsymbol{\theta_D}$ in the discriminator:

$$W(\boldsymbol{\theta_G}, \boldsymbol{\theta_D}) = E[\boldsymbol{D}(I, \boldsymbol{G}(I, D_s)) - \boldsymbol{D}(I, D_d)] + \lambda R(\boldsymbol{\theta_D}) \tag{4}$$

where $\boldsymbol{D}$ is the discriminator network with tunable parameters $\boldsymbol{\theta_D}$, $(I, \boldsymbol{G}(I, D_s)$ and (I, D_d) denote a fake pair (i.e. *image-generated* depth pair) respectively, $R(\boldsymbol{\theta_D})$ is for enforcing 1-Lipschitz constraint of $\boldsymbol{D}$ [26] as,

$$R(\theta_D) = (\left\| \nabla_{\widehat{x}} \boldsymbol{D}_{\boldsymbol{\theta_D}}(\widehat{x}) \right\|_2 - 1)^2 \tag{5}$$

where $\widehat{x} = \epsilon D_d + (1-\epsilon)\boldsymbol{G}_{\boldsymbol{\theta_G}}(I, D_{sd}), \epsilon \sim U[0,1]$.

Accordingly, the depth loss l_D is formulated as the weighted sum of a pixel-wise MSE loss l_D^{MSE} and an adversarial loss l_D^{Adv} as:

$$l_D = \delta l_D^{MSE} + \gamma l_D^{Adv} \tag{6}$$

2) Confidence Loss l_C: The ground truth of confidence for a pixel x is calculated as:

$$C_{gt}(x) = e^{-|D(x) - D_{gt}(x)|} \tag{7}$$

where $D(x)$ and $D_{gt}(x)$ are the predicted depth and ground truth depth of x. We choose the pixel-wise L1 loss to represent the difference between a predicted confidence by the pCNN, $C(x)$, and its ground truth $C_{gt}(x)$. Accordingly, the confidence loss l_C of the pixel-wise L1 loss is calculated as:

$$l_C = \sum_{x \in X} \left| C(x) - C_{gt}(x) \right| \tag{8}$$

By minimizing l_C during the optimization process, the predicted confidence value $C(x)$ can gradually approach to the corresponding ground truth $C_{gt}(x)$. The problem, however, is that the value of $C_{gt}(x)$ also keeps increasing and gradually approaches to one due to the minimization process of the depth loss l_D. As a result, optimization of l_C might become ineffective and "lazy", since the network can output ones for all confidence values which also reduce l_C because of the continuous increase of $C_{gt}(x)$. To overcome this problem, we define another penalty term l_R to prevent the value of $C(x)$ from being trapped at one:

$$l_R = \eta \left| C(x) \right| \quad (0 < \eta < 1) \tag{9}$$

Adding l_R to l_C would not change the optimal solution of l_C, i.e. l_C still reaches its minimum value when $C(x) = C_{gt}(x)$. Meanwhile, adding l_R can increase the gradient of l_C for $C(x)$ that is greater than $C_{gt}(x)$, and in turn can force $C(x)$ to leave the current value more quickly, rather than staying at the current value until $C_{gt}(x)$ reaches this value. Accordingly, the confidence loss l_C is formulated as:

$$l_C = \sum_{x \in X} (\left| C(x) - C_{gt}(x) \right| + l_R) \tag{10}$$

3) Training Data Generation: The pCNN is trained using triplets including an RGB image I_n, the corresponding sparse depth map $D_{s,n}$ and the ground truth dense depth map $D_{d,n}$. Running a monocular visual odometry system in the public visual SLAM benchmarks could provide both sparse depth maps for keyframes and the corresponding ground truth dense depth maps. However, due to the limited number of keyframes in SLAM benchmark datasets, we can hardly collect a sufficiently large number of sparse maps. Another solution is to directly sampling a set of depth values from the ground truth depth $D_{s,n}$. However, such ideal sparse depth map can hardly be obtained by a practical monocular visual odometry system because there exist inevitable noises in sparse depth maps obtained from consecutive view matching and triangulation.

To address this issue, we follow the solution in [14] to synthesize noisy sparse depth maps by simulating mapping noises in a practical visual odometry (i.e. ORB-SLAM) system, which mainly come from the inevitable errors in pixel matching. Specifically, the pixel matching accuracy mainly relies on the image characteristics, and thus pixels within a small local region usually have similar error distributions due to the large similarity of the image characteristic with each other while pixels from regions far apart could have quite different error distributions. Therefore, we select a subset of pixels by the ORB detector from each ground truth depth map to form a candidate sparse depth map, randomly partition a candidate sparse depth map into several regions and define a specific region-wise error distribution for each region. According to the studies in [27], the error distribution of mapping in a monocular SLAM can be approximated using a combination of a Gaussian distribution for inliers and a uniform distribution for outliers. Accordingly, for each local region we randomly choose a variance value ($\sigma \in [0.01, 0.03]$) for the Gaussian distribution, min and max values (we set min=-0.02 and max=0.02) for the uniform distribution. The noise for each pixel is then generated from the corresponding region-wise distribution.

III. Obstacle Avoidance on a Drone

A. Ego Dynamic Space Transform

Trajectory planning without considering the dynamic constraints of a drone and depth estimation error is susceptible to collision. To address this issue, we apply the Ego Dynamic Space (EDS) [28] to embed the dynamic constraints of a drone as well as depth error into a spatial representation. Specifically, in EDS the distances D_{eff} to the obstacles are transformed into effective distances that depend on the robot deceleration constraint, sampling time of the system as well as the depth prediction error as:

$$\begin{aligned} D_{eff} &= D - D_{brake} - D_{error} \\ &= D - (vT - \frac{aT^2}{2}) - D_{error} \end{aligned} \tag{11}$$

where D is the measured distance to the obstacles (i.e. depth map from the pCNN), D_{brake} is the deceleration distance to stop a drone which travels at velocity v using deceleration a in a sampling interval T, and D_{error} is the depth measurement error which is calculated as $D_{error} = -ln(C)$, C is the predicted confidence. The distance D_{eff} represents the maximum distance that a drone can travel safely with a constant velocity v without collision during the period T.

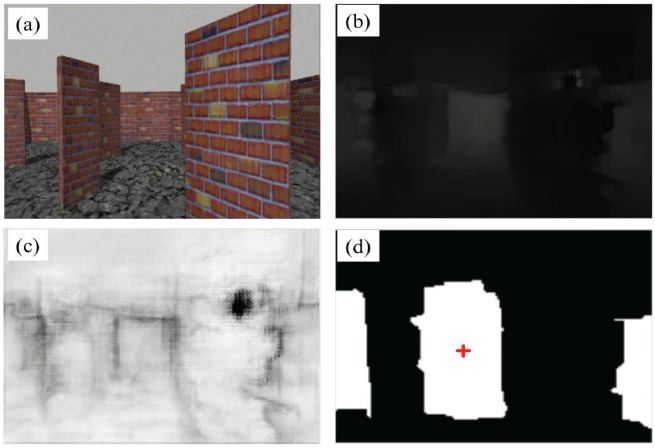


Fig. 3. (a) RGB image, (b) depth map, (c)confidence map, (d)binary depth map with the selected waypoint (cross in red).

B. Waypoint Selection

We quantize D_{eff} into a binary depth map, in which the pixel whose D_{eff} value is greater than 0 is set to "True" and otherwise is set to "False". Then, we group pixels whose values are "True" into clusters via a connected component detection method and those "True" regions correspond to candidate collision-free regions that a quadrotor can transverse. We define r_k and o_k as the radius and centroid of the maximum incircle of the "True" region k respectively. Intuitively, we can choose the centroid o_k as the next waypoint if the radius of r_k achieves the greatest value among all candidate collision-free regions. However, such solution could yield fluctuation of consecutive waypoints and in turn lead to non-smooth trajectories. Therefore, in this work we first check whether there exist a collision-free region along the line of sight (X-axis) whose size is greater that 1.5 times of the drone's size. If yes, we set the center of this region as the next optimum waypoint. Otherwise, we choose a waypoint from candidate collision-free regions which achieves the greatest radius. Fig. 3 illustrates the predicted depth, confidence, candidate collision-free region and the selected waypoints of a simulated scene.

C. Control

We implemented the proposed system and applied it to a drone we built which is controlled using the command $\mathbf{C} = [x, y, z, q_x, q_y, q_z, q_w]^T$. The triple (x, y, z) represents the quadrotor's position and (q_x, q_y, q_z, q_w) are the quaternion of the drone's attitude with respect to its body frame. Once a next optimum command is computed based on the current camera frame, the drone flies under a constant velocity v.

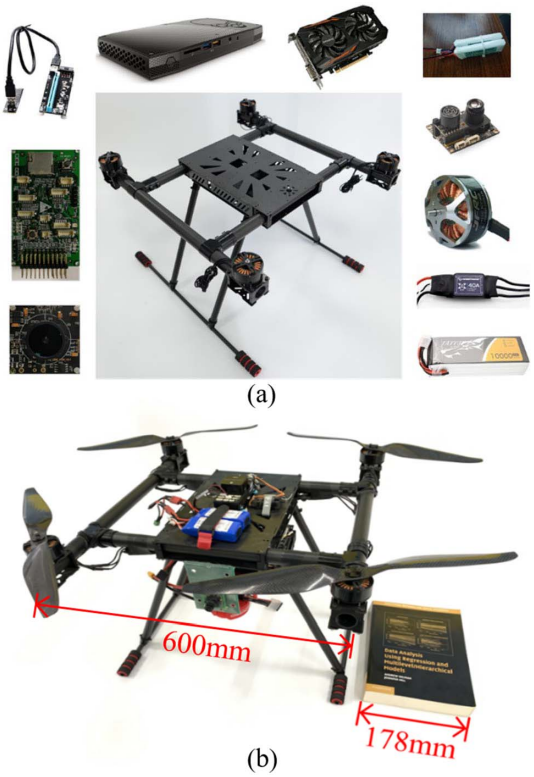


Fig. 4. Components of a quadrotor drone. (a) Different modules in a quadrotor drone. (b) Prototype of the integrated modules. A book with known size is added as a reference to illustrate the size of our drone.

IV. Drone Platform Design

A. Hardware Architecture

Our drone is equipped with an Intel NUC as the main computing platform which is further integrated with an NVIDIA Jetson TX2 through a special designed expansion cable. The Intel NUC core i5, and the TX2 with 8G shared RAM and 256 CUDA cores. The communication between the NUC and the user is handled by Robot Operating System (ROS) through wireless network.

The flight control part of our system is handled by a self-designed flight control board, and we use a PX4 Flow sensor [29] for indoor localization. The open-source autopilot firmware PX4 is used as the flight stack. Based on the Nuttx real-time operating system, PX4 is the first node architecture oriented with a fully multi-threaded modular robotics framework for deeply embedded platforms using a publish/subscribe design pattern [30]. The PX4 flight stack supports the ROS interface and provides native ROS nodes. The bridge between ROS and PX4 has a standard ROS package called MAVROS, which supports the MAVLink protocol and provides both command lines and open-source APIs to establish seamless bilateral communication with the PX4 stack. We optimized this package and succeeded in navigating the quadrotor via a self-built ROS package.

Fig. 4(a) shows the chosen quadrotor frame, namely, QX600, which is integrated with four brushless motors, four propellers, four 40A electronic speed controllers (ESCs), one Intel NUC, one NVIDIA Jetson TX2, one PX4 Flow sensor, one global shutter camera, and one 22.2V LiPo motor battery. Fig. 4(b) shows the prototype assembled with those components. The proposed prototype weighs about 5,059 grams and has a 600 mm wheelbase. It can typically fly for approximately 20 minutes.

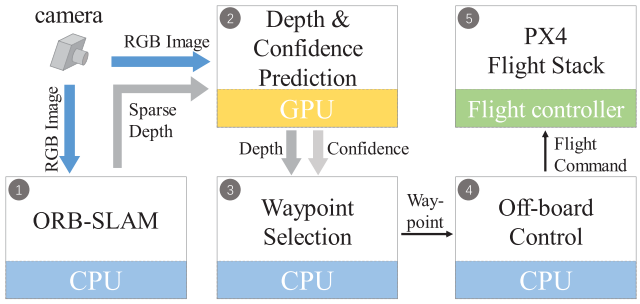


Fig. 5. Software Architecture.

B. Software Architecture

The proposed software architecture consists of five steps, as illustrated in Fig. 5 The first step is running on the CPU core, on which an ORB-SLAM based sparse mapping is performed to output sparse depth maps. The second step is running on the GPU core, on which our pCNN takes the captured RGB image and its corresponding sparse map as input to produce a dense depth image and a confidence map. The output depth and confidence maps are then published in the ROS for the following obstacle avoidance steps. The third step is running on the CPU core to perform waypoint selection and the selected waypoint is transformed into a control command which is published to the flight controller via the UART interface in the fourth step. The last step is an independent flight control module, which controls the flight of the drone and transforms the received flight command into the drone's location and orientation.

V. Experimental Results

In this section, we first evaluate the performance of depth and confidence prediction in terms of accuracy and runtime on both PC and drone platform. Both quantitative and qualitative results obtained by our model and comparison with the state of the art on standard benchmark datasets are provided. After that, we evaluate the performance of our onboard obstacle avoidance system in both simulated environments and real indoor environments.

A. Performance of Depth and Confidence Prediction

1) Implementation Details: We implemented our network using Tensorflow [31]. The entire network, including depth predictor and discriminator, was initialized as random filters sampled from a normal distribution with zero mean and 0.01 variance. For indoor scenario, we trained our network using the training set of NYU Depth v2 [32]. Specifically, the dataset consists of 464 scenes, captured using a Microsoft Kinect, with an official split consisting of 249 training and

TABLE I
COMPARISON RESULTS OF DEPTH PREDICTION ON NYU DEPTH-V2

Method	Error		Accuracy			Runtime(ms)		
	Rel.	RMS	δ_1	δ_2	δ_3	TITANX	TX2	1050Ti
Laina-ResNet50[7]	0.127	0.573	0.811	0.953	0.988	58	228	39
Laina-ResNet18[7]	0.265	0.775	0.606	0.855	0.948	14	73	22
CNN-SLAM[13]	0.131	0.537	0.846	0.964	0.991	58	228	39
Yang et al.[14]	0.064	0.243	0.966	0.996	0.999	39	467	87
Ours Original	0.206	0.683	0.668	0.899	0.971	10	76	20
Ours+Sparse	0.047	0.209	0.973	0.995	0.999	11	79	19
Ours+l_D^{Adv}	0.199	0.628	0.708	0.916	0.975	10	76	20
Ours+Sparse+l_D^{Adv}	0.037	0.172	0.981	0.997	0.999	12	84	22

TABLE II
COMPARISON RESULTS OF DEPTH PREDICTION ON KITTI

Method	Error		Accuracy			Runtimes (TITANX, ms)
	Rel.	RMS	δ_1	δ_2	δ_3	
Eigen et al.[6]	0.190	7.156	0.692	0.899	0.967	3604
Liu et al. [12]	0.217	6.986	0.647	0.882	0.961	-
Kuznietsov et al. [10]	0.113	4.621	0.862	0.960	0.986	58
Ours	0.125	5.752	0.869	0.956	0.980	12

TABLE III
COMPARISON RESULTS OF CONFIDENCE PREDICTION ON NYU DEPTH-V2

Method	Error		Accuracy		
	Rel.	RMS	δ_1	δ_2	δ_3
Ours Original	1.352	0.290	0.668	0.692	0.813
Ours+Sparse	0.103	0.101	0.917	0.975	0.989
Ours+l_D^{Adv}	1.140	0.266	0.527	0.732	0.842
Ours+Sparse+l_D^{Adv}	0.085	0.085	0.948	0.983	0.992

215 test scenes. We down-sampled the original frames of size 640×480 pixels to 1/2 resolution as input to our network and then resized the frames to 384×288 via zero padding before convolutional layers. We sampled equally spaced frames from each training sequence, resulting in approximately 12k frames. We further augmented the selected frames using the same strategies as used in Eigen *et al.* [5] and Eigen and Fergus [6], yielding approximately 95k frames with its corresponding depth ground truth for training our network. To demonstrate our method in outdoor environment, we trained our model using 20K training images from the Virtual KITTI dataset [33] rather than directly using the training set of KITTI [34]. This is because our model, which is trained using the adversarial loss, requires the dense depth map as the ground truth. However, the KITTI dataset only provides sparse depth ground truth. We evaluate the four methods on 697 test images of the KITTI dataset. We trained both indoor and outdoor CNN models with a batch size of 4 for approximately 36 epochs. The starting learning rate is 10^{-4} for all layers, which is gradually reduced by 10% of the previous value every 8 epochs when plateaus are observed. We experimental test a list of parameter settings for α, β, δ, γ and η in Eqs.(3) (6) and (9), and chose the setting 1.0, 0.5, 1.0, 0.0001, 0.5 as it could make the total loss converge smoothly and steadily to a reasonable value and meanwhile balance the performance of both depth and confidence estimation.

We use the same metrics which have been widely used in prior studies [7] for quantitative evaluation, which include the absolute relative error (REL.), rooted mean square error (RMSE) and accuracy. Specifically, the accuracy metrics are defined as:

$$\delta_n = \frac{\mathbf{card}(\left\{ d_i : max(\frac{d_i}{d_i^{gt}}, \frac{d_i^{gt}}{d_i}) < 1.25^n \right\})}{\mathbf{card}(\{d_i\})} \quad (n=1,2,3) \quad (12)$$

where d_i and d_i^{gt} are estimated depth and ground truth of pixel i, and $\mathbf{card}(\cdot)$ is cardinality of a set. A higher δ_n indicates better predicition.

2) Performance of Depth Prediction: We evaluate our depth prediction method as well as the comparison methods on the NYU Depth-v2 test set. Table I compares the performance of our depth prediction method (denoted as "Ours+sparse+l_D^{Adv}") with two versions of the state-of-the-art method [7] (i.e. Laina-ResNet50 and Laina-ResNet18), CNN-SLAM [13], our previous method (Yang *et al.*) [14] as well as three variants of our methods (denoted as "Ours Original", "Ours+Sparse" and "Ours+l_D^{Adv}") on the NYU Depth-v2 test set and KITTI test set. To date, Laina's method using the ResNet50 as the basic feature extraction network, denoted as Laina-ResNet50, achieves the best performance on most publically available datasets including NYU Depth-v2. In comparison, Laina-ResNet18 is ~3X faster than Laina-ResNet50 at a cost of non-trivial performance degradation. For "Ours Original", the network uses only one branch to process an RGB input image and the adversarial loss is not used during the training process. For "Ours+Sparse", the network contains two input branches while the adversarial loss is not used for training and for "Ours+l_D^{Adv}" the adversarial loss is utilized for training while only a single input RGB branch is employed. According to the results shown in Table I, five observations can be made. First, when comparing "Ours Original" with "Ours+Sparse" we observe that adding sparse depth maps could significantly improve the performance, e.g.

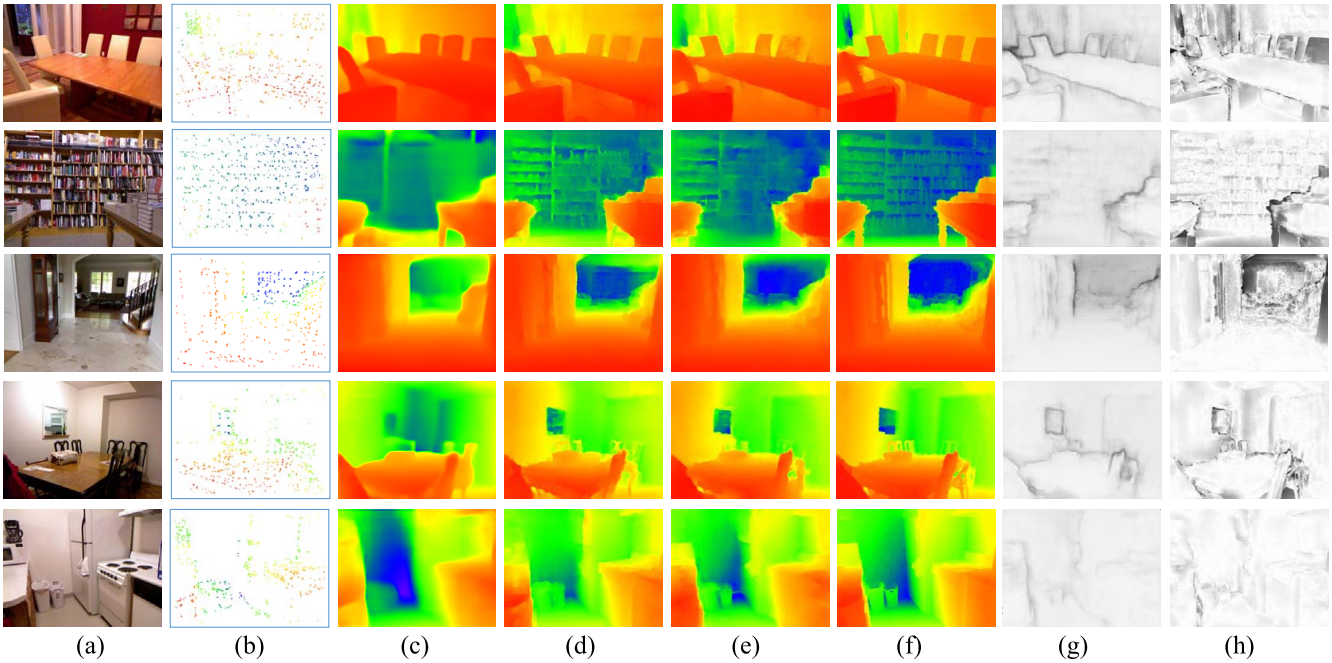


Fig. 6. (a) RGB images, (b) sparse depth maps, (c) predicted depth maps of Laina et al. method, (d) predicted depth maps of Yang et al. method, (e) predicted depth maps of our method, (f) ground truth depth maps, (g) predicted confidence maps of our method, (h) ground truth confidence maps.

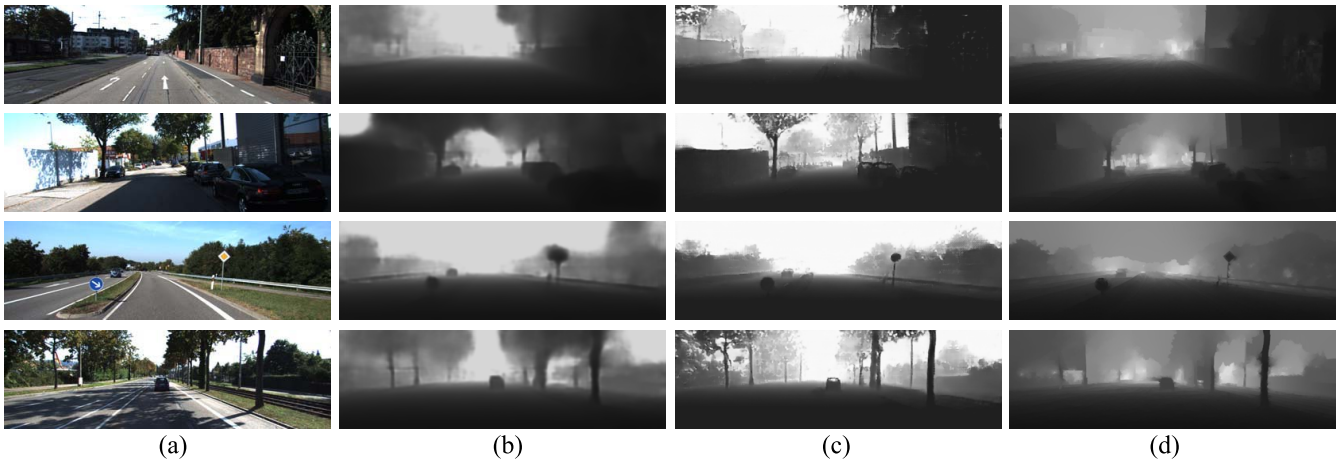


Fig. 7. (a) RGB images of KITTI, (b) predicted depth maps of Kuznietsov et al. method, (c) predicted depth maps of our method, (d) interpolated ground truth depth maps.

77.2% reduction in the relative error (REL.), demonstrating the effectiveness of using a sparse depth map to guide dense depth inference. Second, by comparing "Ours Original" with "Ours+l_D^{Adv}" we could observe that adding an adversarial loss could also improve the depth prediction performance, but the performance improvement is smaller than adding sparse depths. Third, the two proposed methods (i.e. GAN and sparse depths) could achieve complementary improvements, and thus "Ours+Sparse+l_D^{Adv}" achieves better performance than "Ours+Sparse" and "Ours+l_D^{Adv}". Fourth, by comparing "Ours+Sparse+l_D^{Adv}" with the two versions of Laina's method and our previous work [14] we observe that our method could achieve much higher depth estimation accuracy than all the three comparison methods. The last three columns show the runtime of all the methods on three platforms, i.e. Nvidia TITANX GPU with 12GVRAM and 3584 CUDA cores, TX2 with 8G shared RAM and 256 CUDA cores, 1050Ti with 4GVRAM and 768 CUDA cores. The runtime results show that "Ours+Sparse+l_D^{Adv}" is 4.4X, 2.7X and 1.8X faster than Laina-ResNet50 on the three platforms respectively, 3.3X, 5.6X and 4.0X faster than our previous work on the three platforms respectively, and achieves a similar speed as Laina-ResNet18.

Table II shows the comparison results of depth prediction on KITTI dataset. Our method achieves superior performance to [6] and [12] for all metrics. Compared with [10], our method is slightly worse in terms of accuracy, while is 4.8X faster than [10]. We think the reasons of slightly worse

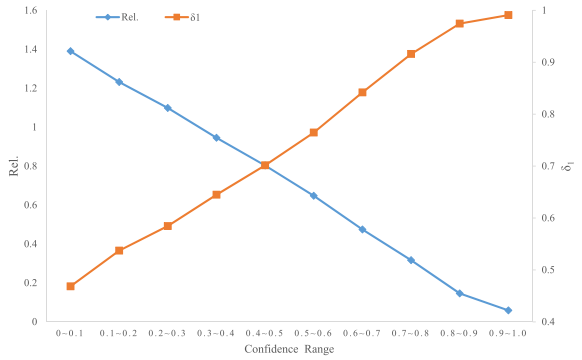


Fig. 8. Average depth estimation error and accuracy on the test set when varying the confidence range.

(a) (b) (c)

Fig. 9. 3D point cloud of two exemplar test scenes based on (a) raw depth prediction, (b) after outlier removal using the predicted confidence, and (c) ground truth point cloud.

accuracy of our method than [10] are three aspects: 1) The distribution/property of the training data used for our model is different from that of the test data. In comparison, [10] trained their model directly on the KITTI training data which is highly similar with KITTI testing data; 2) We used a shallow network architecture (i.e. ResNet-18 as our backbone) to achieve a high efficiency on drone platform, while [10] used ResNet-50 as the backbone and in turn yields higher computational complexity; 3) The test data of KITTI only provides sparse depth obtained from Lidar as ground truth, which cannot provide a complete evaluation for dense depth prediction of our method due to majority of missing ground truth depth values.

Fig. 6(e) and (f) illustrate the predicted indoor depth map of our method and the corresponding ground truth depth map of five exemplar scene frames. It can be observed that the depth maps predicted by our method exhibit noteworthy visual quality that is close to the ground truth maps. Fig. 7 displays the visualization results of some exemplar images in KITTI.

3) Performance of Confidence Prediction: We also evaluate the performance of our confidence prediction in terms of error, accuracy and the effectiveness for excluding outliers on the NYU Depth-v2 test set. Table III presents the error and accuracy of confidence prediction on the NYU dataset. Results show that by utilizing both sparse depth maps and adversarial learning, "Ours+Sparse+l_D^{Adv}" achieves superior performance to the other variants of our methods for all metrics. The last two columns of Fig. 6 (e) and (f) show the predicted confidence maps and the ground truth confidence respectively. Generally, the confidence maps obtained by our method are similar to the ground truth confidence maps.

To analyze the effectiveness of our predicted confidence for excluding outliers, we set a confidence threshold with an increasing number of values ranging from 0.0 to 0.95. For a predicted depth map, pixels whose confidence values are smaller than the threshold are discarded as outliers. Fig. 8 shows that, as the threshold increases, the depth estimation error decreases and accuracy improves. These results imply that the output confidence is consistent with depth estimation accuracy, i.e. pixels with high confidence values also have high depth estimation accuracy, demonstrating the effectiveness of our network for confidence prediction. In Fig. 9, we also show the visualization of 3D point cloud when the confidence threshold is set as 0.6 for outlier removal as it could provide the most regions with reliable depth prediction and meanwhile remove as many inaccurate depth regions as possible. The raw depth prediction from our network could contain large errors for some pixels, yielding obvious depth discontinuities in the 3D point cloud as shown in Fig. 9(a). Excluding points whose confidence is lower than 0.6 from the 3D point cloud effectively removes those outliers (as denoted by white regions in Fig. 9(b)), producing a cleaner visualization result. As a result, the 3D point cloud after outlier removal resembles the result well based on back-projection of ground truth depth maps into 3D space.

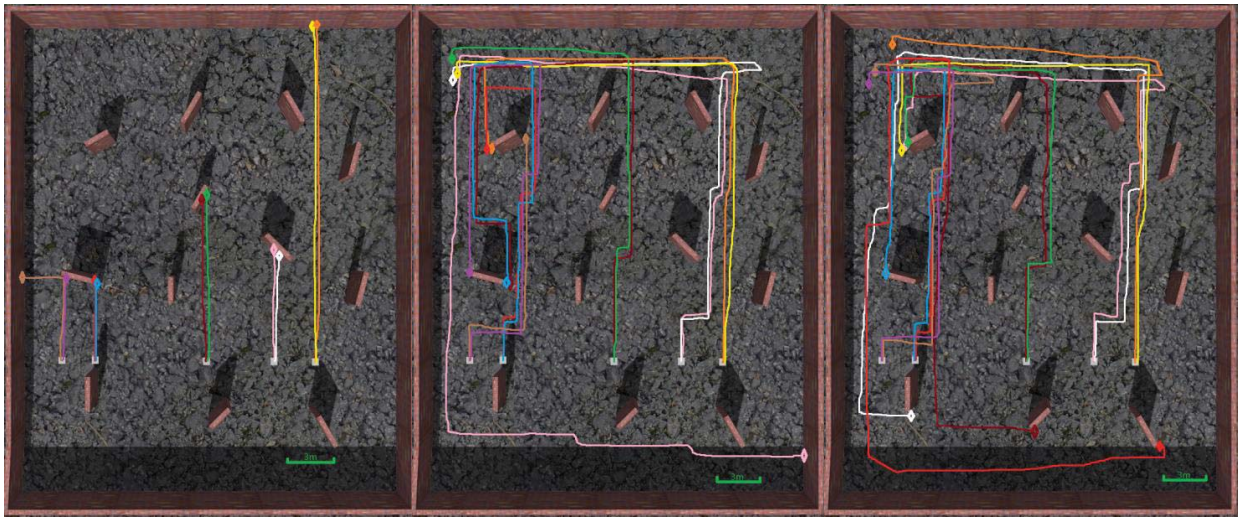
TABLE IV

PERFORMANCE OF OUR METHOD IN THE SIMULATED ENVIRONMENTS

	Laina (ResNet18)	Ours+Sparse+l_D^{Adv} w/o Confidence	Ours+Sparse+l_D^{Adv} w/ Confidence
Mean crash time (s)	30.2	168.3	200.4
Mean distance traveled (m)	7.088	28.28	31.792

B. Performance of Obstacle Avoidance in Simulated Environments

We generated a simulated world in Gazebo to test the performance of our method. The simulated world is an enclosed room with pole-like wall segments placed in the center as shown in Fig. 10. In this evaluation, we quantitatively test the performance of obstacle avoidance of three methods: 1) Laina-ResNet18 which achieves a similar runtime performance as Ours with a much lower depth prediction accuracy, 2) Ours without using the predicted confidence for obstacle avoidance and 3) Ours with the predicted confidence. For all three methods, we set off the drone at the same starting point; the simulated drone is spawned in different locations in the world and allowed to fly till it crashes. We recorded the flying time and traveled distance of each flight before collision. The longer flying time and traveled distance, the better the performance. For each method we ran the test 10 times to calculate the mean crash time and mean distance traveled. Table IV shows the average results of the three methods. By comparing the first



(a) (b) (c)

Fig. 10. Top-down view of the simulated world with trajectories from simulated flights. (a) Depth prediction with Laina-ResNet18, (b) depth prediction with our method without using predicted confidence maps, (c) depth prediction with our method using the predicted confidence maps for obstacle avoidance.

Fig. 11. Automated obstacle avoidance of a drone in the indoor lab environment (10/10 successful flights).

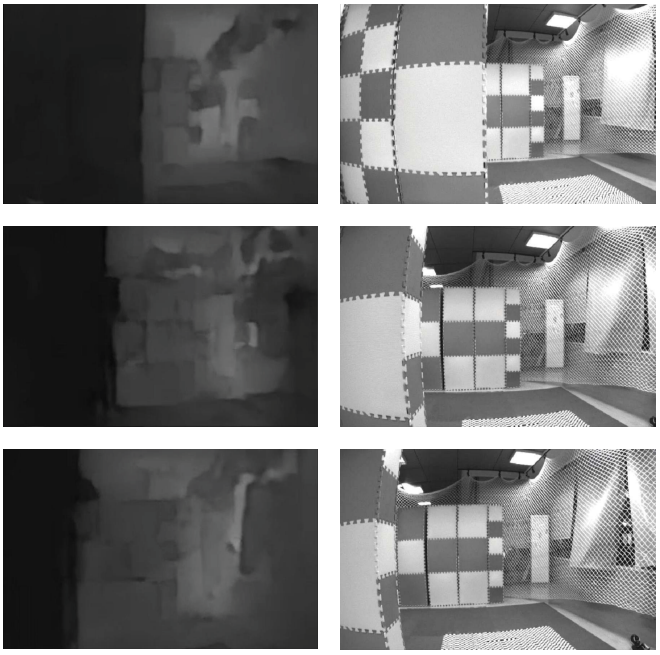


Fig. 12. Depths and the corresponding raw images from the perspective of the drone during the obstacle avoidance process in a real environment.

two columns of Table IV we observe that the mean crash time and the distance traveled based on "Laina-ResNet18" are much shorter than those of "Ours+Sparse+l_D^{Adv} w/o confidence", denoting that more accurate depth prediction could yield better performance for obstacle avoidance. By comparing the last two columns we observe that, the predicted confidence map could better facilitate collision avoidance and in turn lead to longer traveled time and distance. All the flying trajectories of the drone in the environment are displayed in Fig. 10. Small squares and diamonds denote the starting points and ending points respectively.

C. Performance of Obstacle Avoidance in Real Indoor Environments

We performed indoor experiments in our lab environment, where we put up foam pads as obstacles. The drone needs to take off and flying autonomously across two obstacles with proper flying angles through a narrow space until landing. Fig. 11 shows a snapshot of our indoor environments with obstacles and its layout from a top view. We also show a trajectory of the drone while flying through all the designed environment. Since we are unable to equip our small drone with an accurate positioning device, we cannot obtain the exact flying trajectory for the evaluation. We evaluate the success rate of the drone obstacle avoidance to demonstrate the effectiveness and feasibility of our method in real indoor scenarios. An experiment is considered a success if the drone successfully steered its way through all obstacles. We fly our drone ten times based on our method and it did in all 10 of the experiment resulting in a 100% success rate. Due to the narrow experimental environment and according to the drone's weight, we set the average flying speed of the drone to 0.6m/s through obstacle avoidance experiment. It is worth noting that our indoor scenario, which consists of many repetitive patterns, is quite different from the training images

of depth prediction network, incurring greater difficulty in depth prediction and obstacle avoidance. A high success rate of our method demonstrates the good generalization ability and its feasibility to practical unseen environments. Fig. 12 shows depths and the corresponding raw images from the perspective of the drone during the obstacle avoidance process in the real environment.

VI. Conclusion

This paper presents a monocular depth prediction and obstacle avoidance drone system which can run real-time on a drone platform and achieves the state-of-the-art accuracy. The key enabling technique of our system is a lightweight pCNN which integrates both sparse depths and RGB images for accurate dense depth and confidence prediction. The complex correlation among sparse depths, raw pixel intensities, true depths and prediction confidence are learned by minimizing three loss functions: 1) an adversarial loss for ensuring high similarity between the geometric structure of true depths and predicted depths, 2) a pixel-wise MSE loss for ensuring absolute depth values being accurate, and 3) a pixel-wise L1 loss with a penalty term for ensuring correct predicted confidence values. Extensive experimental results demonstrate that our lightweight CNN runs 1.8X∼5.6X faster than the state-of-the-art method and achieves a much higher depth prediction accuracy. We employed the lightweight depth pCNN to our drone system for obstacle avoidance and evaluated its performance in both simulated and real environments. Experimental results show that, compared to the state-of-the-art network which has a similar runtime speed as ours, our depth CNN could effectively reduce the collision probability of collision. In addition, the predicted confidence map from our method could further improve the success rate of avoiding obstacles.

In our future work, we will explore the usage of other cost-effective and lightweight active depth sensors, which could provide sparse and uniformly distributed depth maps, and integrate these inputs into our network. In this study, we used a simple motion planning scheme for avoiding obstacles. In our future work, we will develop more advanced motion planning methods to further improve the performance.

References

[1] K. Bipin, V. Duggal, and K. M. Krishna, "Autonomous navigation of generic monocular quadcopter in natural environment," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 1063–1070.

[2] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 593–600.

[3] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, "CNN-based single image obstacle avoidance on a quadrotor," in *Proc. IEEE Int. Conf. Robot. Autom.*, May/Jun. 2017, pp. 6369–6374.

[4] I. Lenz, M. Gemici, and A. Saxena, "Low-power parallel algorithms for single image based obstacle avoidance in aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 772–779.

[5] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2366–2374.

[6] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2650–2658.

[7] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks" in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 239–248.

[8] R. Garg, V. B. G. Kumar, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 740–756.

[9] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. CVPR*, Jul. 2017, vol. 2, no. 6, pp. 6602–6611.

[10] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6647–6655.

[11] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2800–2809.

[12] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 5162–5170.

[13] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jul. 2017, pp. 6243–6252.

[14] X. Yang *et al.*, "Monocular camera based real-time dense mapping using generative adversarial network," in *Proc. ACM Multimedia Conf. Multimedia Conf.*, 2018, pp. 896–904.

[15] H. Luo, Y. Gao, Y. Wu, C. Liao, X. Yang, and K.-T. Cheng, "Real-time dense monocular SLAM with online adapted depth prediction network," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 470–483, Feb. 2019.

[16] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5574–5584.

[17] H. Tian and F. Li, "Semi-supervised depth estimation from a single image based on confidence learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Processing (ICASSP)*, May 2019, pp. 8573–8577.

[18] H. Wang, Q. Zhang, Y. Wang, and H. Hu, "Structured probabilistic pruning for convolutional neural network acceleration," Sep. 2017, *arXiv:1709.06994*. [Online]. Available: https://arxiv.org/abs/1709.06994

[19] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[21] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, p. 155, 2009.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," Nov. 2015, *arXiv:1511.06434*. [Online]. Available: https://arxiv.org/abs/1511.06434

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017, *arXiv:1701.07875*. [Online]. Available: https://arxiv.org/abs/1701.07875

[26] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5769–5779.

[27] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2014, pp. 15–22.

[28] J. Minguez, L. Montano, and O. Khatib, "Reactive collision avoidance for navigation with dynamic constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, Sep./Oct. 2002, pp. 588–594.

[29] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2013, pp. 1736–1741.

[30] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 6235–6240.

[31] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16. 2016, pp. 265–283.

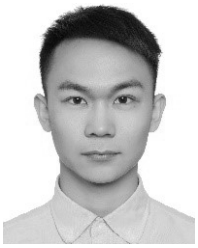
[32] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2012, pp. 746–760.
[33] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proc. CVPR*, Jun. 2016, pp. 4340–4349.
[34] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

Xin Yang (M'15) received the Ph.D. degree from the University of California, Santa Barbara, in 2013. She was a Post-Doctoral Researcher with the Learning-based Multimedia Laboratory, UCSB, from 2013 to 2014. She is currently an Associate Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology. She has published over 40 technical articles, including TPAMI, TMI, TVCG, TMM, MedIA, CVPR, ECCV, and MM, coauthored two books and holds 30 U.S. Patents. Her research interests include monocular simultaneous localization and mapping, augmented reality, and medical image analysis. She is a member of ACM.

Jingyu Chen received the B.E. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017, where he is currently pursuing the master's degree with the School of Electronic Information and Communications. His research interests include monocular simultaneous localization and mapping and deep learning.

Yuanjie Dang was born in Shannxi, China, in 1993. He received the B.Sc. degree in electronic science and technology from Zhejiang Sci-Tech University, Hangzhou, China, in 2015. He is currently pursuing the Ph.D. degree in control science and engineering with the Zhejiang University of Technology. His current research interests include UAV visualinertial navigation, UAV autonomous flight control, and visual tracking.

Hongcheng Luo received the B.E. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2016, where he is currently pursuing the master's degree with the School of Electronic Information and Communications. He has published two articles on ACM MM 2017. His research interests include monocular simultaneous localization and mapping and augmented reality.



Yuesheng Tang was born in Fujian, China, in 1995. He received the B.Sc. degree in electrical engineering and automation from the Zhejiang University of Technology, Hangzhou, China, in 2017, where he is currently pursuing the M.Sc. degree in control science and engineering. His current research interests include UAV vision-based navigation and machine learning.

Chunyuan Liao received the Ph.D. degree from the University of Maryland, College Park, in 2008. He was a Research Scientist with the Fuji Xerox Palo Alto Laboratory, Silicon Valley, where he was granted significant achievement awards three times and ACM conference awards twice due to his outstanding research in Augmented Reality. He has published over 60 technical articles, including ACM Transactions on CHI, UIST, ICCV, ACM MM, and CHI. He holds more than 10 U.S. Patents. He founded HiScene, a leading Chinese Augmented Reality technology provider in 2012. He is one of the national distinguished experts in Chinese government's "The Thousand Talents Plan".

Peng Chen was born in Zhejiang, China, in 1981. He received the B.Sc. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2003 and 2009, respectively. He was a Visiting Scholar with the University of California at Santa Barbara, Santa Barbara, USA, from 2015 to 2016. He is currently a Professor with the Zhejiang University of Technology. His major research fields include computer vision and signal processing.

Kwang-Ting (Tim) Cheng (S'88–M'88–SM'98–F'00) was with the AT&T Bell Laboratories from 1988 to 1993. Before joining HKUST in 2016, he was a Professor of electrical and computer engineering (ECE) with the University of California, Santa Barbara, where he has been since 1993. He is currently the Chair Professor and the Dean of engineering with HKUST. An internationally leading researcher with rich experience in fostering cross-disciplinary research collaboration, his research interests and contributions include VLSI testing and design verification, design automation of electronic and photonic systems, computer vision, and medical image analysis. He had previously served as the Director of the U.S. Department of Defense Multidisciplinary University Research Initiative (MURI) Center for 3-D Hybrid Circuits. He has published more than 450 technical articles, coauthored five books, held 12 U.S. patents, and transferred several of his inventions into successful commercial products.