drones

MDPI

Article

End-to-End Nano-Drone Obstacle Avoidance for Indoor Exploration

Ning Zhang *, Francesco Nex, George Vosselman and Norman Kerle

ITC Faculty Geo-Information Science and Earth Observation, University of Twente, 7522 NH Enschede, The Netherlands; f.nex@utwente.nl (F.N.); george.vosselman@utwente.nl (G.V.); n.kerle@utwente.nl (N.K.)
* Correspondence: n.zhang@utwente.nl

Abstract: Autonomous navigation of drones using computer vision has achieved promising performance. Nano-sized drones based on edge computing platforms are lightweight, flexible, and cheap; thus, they are suitable for exploring narrow spaces. However, due to their extremely limited computing power and storage, vision algorithms designed for high-performance GPU platforms cannot be used for nano-drones. To address this issue, this paper presents a lightweight CNN depth estimation network deployed on nano-drones for obstacle avoidance. Inspired by knowledge distillation (KD), a Channel-Aware Distillation Transformer (CADiT) is proposed to facilitate the small network to learn knowledge from a larger network. The proposed method is validated on the KITTI dataset and tested on a Crazyflie nano-drone with an ultra-low power microprocessor GAP8. This paper also implements a communication pipe so that the collected images can be streamed to a laptop through the on-board Wi-Fi module in real-time, enabling an offline reconstruction of the environment.

Keywords: drone; obstacle avoidance; computer vision; depth estimation; transformer; knowledge distillation; edge computing; Crazyflie

Citation: Zhang, N.; Nex, F.; Vosselman, G.; Kerle N. End-to-End Nano-Drone Obstacle Avoidance for Indoor Exploration. *Drones* **2024**, *8*, 33. https://doi.org/10.3390/drones8020033

Academic Editor: Diego González-Aguilera

Received: 26 December 2023
Revised: 17 January 2024
Accepted: 20 January 2024
Published: 24 January 2024

1. Introduction

Drones play an important role in exploration tasks. In particular, nano-sized drones are suitable for exploring narrow and cluttered environments after disasters because of their small size and relative affordability [1]. Flying autonomous drones in scenarios where GNSS is not available is a challenging research topic. Some research integrates multiple sensors, such as high-quality stereo cameras [2,3] and LiDAR [4–7] for drone navigation. These methods integrate with on-board SLAM algorithms and a map of the environment can be constructed [8,9], which is useful for planning the rescue mission after a disaster. However, this type of method requires drones with large payloads [10] and running on-board SLAM algorithms is not possible due to their weak processors. As computer vision technology evolves, learning-based methods using monocular vision emerge. Given an input image a convolutional neural network (CNN) can be trained to directly output control commands [11–13]. Since this type of method cannot control the drone's next waypoints and relies on black-box decision-making, it is more reasonable to navigate and avoid obstacles based on intermediate depth maps predicted by CNNs [14–16]. The advantages of using depth maps for drone navigation are two-fold: (i) A depth map intuitively represents the distance from each object to the viewpoint in the scene and is ideal for navigating a drone and selecting the next waypoint if needed. (ii) Some recent CNN-based depth estimation methods leverage the self-supervised training strategy and do not require labeled data for training. In view of these advantages, the focus of this paper is also on the use of the depth map estimated by a CNN for the obstacle avoidance of nano-drones.

The target platforms of the aforementioned methods were not nano-drones, and they used high-performance graphics processing units (GPUs), which are not available in nano-drones. The nano-drone platform used in this paper is Bitcraze's Crazyflie, which uses GAP8 as its processor and has only 22.65 GOPS of processing power and 512 KB of RAM. There is not enough memory to store two 324 × 324 RGB images and other files such as model weights. To our knowledge, this paper is the first to implement CNN-based depth estimation networks on such a nano-drone with extremely limited computational capacity. The contribution of this paper can be summarized as follows.

- In order to reduce drone memory usage, this paper explores running depth estimation networks on single-channel grayscale images. The experimental results on the KITTI dataset show that self-supervised depth estimation using grayscale images is feasible. A single grayscale image saves two-thirds of the storage space compared to using a single RGB image.
- The remaining space is still insufficient for the storage and inference of small models such as Lite-Mono [17]. Therefore, a lightweight depth estimation framework DDND is proposed that has few parameters (310 K). To compensate for the limited learning capacity of the small network, knowledge distillation is introduced and a Channel-Aware Distillation Transformer (CADiT) is proposed to make the student model explore important information in different feature channels from the teacher, thus enhancing the knowledge distillation. The effectiveness of the method is validated on the KITTI dataset.
- The proposed model is deployed on the nano-drone platform Crazyflie, and it runs at 1.24 FPS on a GAP8 processor to avoid obstacles in real environments. The code will be released on the project website https://github.com/noahzn/DDND (accessed on 17 January 2024).
- Since a map of the environment is useful for rescue missions and considering that it is not possible to run reconstruction algorithms onboard nano-drones, this paper implements data communication from the drone to a laptop and presents a pipeline for offline reconstruction of the environment.

The rest of the paper is organized as follows. Section 2 reviews some related research work. Section 3 describes the proposed method in detail. Section 4 elaborates on the experiments and discussions. Section 5 introduces the proposed pipeline for reconstructing the environment offline on a laptop. Section 6 concludes the paper.

2. Related Work

2.1. Obstacle Avoidance of Nano-Drones

Nano-drones can be equipped with small laser rangers [1,18–20] or sonars [21] to avoid obstacles at short distances. Some research focused on optical flow estimation using a dedicated optical flow sensor [22] or cameras [23,24]. With the development of edge computing devices, obstacle avoidance using CNN-based methods is beginning to make its mark on edge computing platforms like PULP [25] and Crazyflie. Image-based methods can provide rich cues of a scene, but some methods directly regressed control commands from a single image, and they did not utilize the geometric information of scenes. For example, Kouris et al. proposed a regression network to output steering angles to control the drone [11]. Similarly, DroNet used a CNN to output both the steering angle and the probability of collision to control the forward velocity. It was trained using car driving images and all the images were annotated as "collision" or "no collision" [12]. Gandhi et al. created a UAV crash dataset and taught their drone to "go left", "go straight", or "go right" using a shallow network [26]. Zhilenkov et al. deployed a similar system for autonomous drone navigation in forests [13]. In comparison, methods using depth maps are favorable [16] because controlling commands or path planning can be built upon this intermediate step and the depth maps can be used for other tasks such as scene reconstruction and exploration. Yang et al. proposed a probabilistic CNN to predict monocular depth and guide the drone to avoid obstacles [14]. Chakravarty et al. trained

a supervised depth estimation network and controlled the drone based on the behavior arbitration algorithm [15]. However, reliable depth estimation is computationally expensive, and the mentioned methods ran on larger platforms or off board. It is not feasible to directly deploy such models on nano-drone platforms. Additionally, the available memory is also a problem. For example, the drone platform used in this work has 512 K L2 (second-level RAM) memory, and this is where the chip executes code and stores captured images. The above methods all processed three-channel RGB images, and it will take more than half the memory ($324 \times 324 \times 3 = 315$ kB) just to store an RGB image, let alone store other code and model files. Instead, using a single-channel grayscale image can save two-thirds of the memory used by an RGB image. This allows us to use a larger CNN model for better performance.

2.2. Efficient Monocular Self-Supervised Depth Estimation

In recent years, single-image depth estimation (SIDE) has attracted considerable attention from researchers with the development of deep learning. SIDE models using supervised training regress pixel-wise depth values from labeled data. As it requires additional effort to annotate the data, self-supervised depth estimation (SSDE) stands out and predicts depth from label-free monocular videos [27]. Subsequent work improved prediction accuracy by introducing multi-task learning [28,29], adding uncertainty constraints [30,31], or using more powerful deep learning architectures [32–34]. Some recent methods have pursued a balance of model accuracy, speed, and size, which is also the focus of this paper. Fastdepth [35] adopted MobileNet [36] as the encoder and achieved fast inference speed on embedded systems. However, this model was designed for supervised training. R-MSFM [37] designed a feature modulation module to learn multi-scale depth features and reduced model size by controlling the encoder's layers. Lite-Mono [17] is a hybrid CNN and Transformer architecture, which is capable of extracting both enhanced local features and global contexts and achieving a state-of-the-art performance. It has a good trade-off between accuracy and model size (3.1 M parameters). Nonetheless, such a small model still exceeds the storage space of the GAP8 processor. To obtain a much smaller model, the Lite-Mono network is streamlined and a model with 310 K parameters is obtained in this paper. This results in the problem that the learning ability of this small model would be limited. For the purpose of improving the learning ability of the model, this paper introduces knowledge distillation.

2.3. Knowledge Distillation for Depth Estimation

In a typical knowledge distillation (KD) framework, a larger teacher model transfers its knowledge to a smaller student model. The common ways to perform KD are through soft labels [38,39] or intermediate feature matching [40–43]. KD has been applied to depth estimation tasks to boost lightweight models. Wang et al. [44] used ResNet-101 [45] as the teacher and MobileNet as the student and set up distillation between decoders of the two networks. Hu et al. [46] improved knowledge distillation with auxiliary data. Pilzer et al. [47] also explored KD for depth estimation, but their method required stereo image pairs for training. However, the student models used in the above-mentioned research were still too heavy to be deployed on a GAP8. Some recent papers have pointed out that KD may have difficulty optimizing the student model and may achieve unsatisfactory results if there is a large learning capacity gap between teacher and student [48,49]. Inspired by some KD methods for classification and semantic segmentation tasks [42,43,50], this paper proposes the CADiT module to encourage the student to pay attention to geometric cues from the teacher's feature channels, thus improving the KD process.

3. Method

The proposed distilled depth for nano-drones (DDND) is shown in Figure 1 and explained in detail in this section. First, the architecture of the network and the depth estimation training scheme are presented. Then, the KD scheme including the proposed

CADiT is demonstrated. The last subsection introduces the control method using the generated depth map.

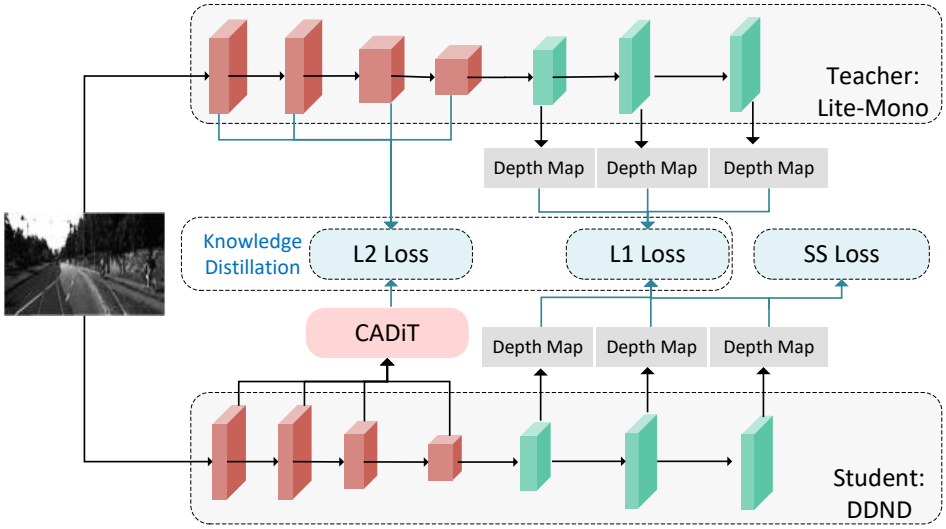

Figure 1. Overview of the proposed DDND. In addition to the self-supervised (SS) loss used in the SSDE training scheme, L2 loss and L1 loss are used to distill the teacher's knowledge into the student's encoder and decoder, respectively. The proposed CADiT is introduced in Section 3.3.

3.1. Network Structures

To make the model deployable on the GAP8 chip, the encoder of Lite-Mono [17] is streamlined to reduce the number of trainable parameters. As shown in the upper part of Figure 2, the student model (DepthNet) is an encoder–decoder network, with four stages in the encoder to extract features. Its decoder concatenates features from the encoder and outputs inverse depth maps at three scales. The channel numbers of the encoder in Lite-Mono are [48, 48, 80, 128], while the student model used in this paper has channel numbers $[C1, C2, C3, C4] = [32, 32, 64, 80]$. The network takes one-channel gray images as input. The same dilation rates of Lite-Mono are used in the DDND network, and the total parameters are reduced from 3.1 M to 310 K. The PoseNet is the same pre-trained ResNet-18 used in [27], and it is not needed after training.

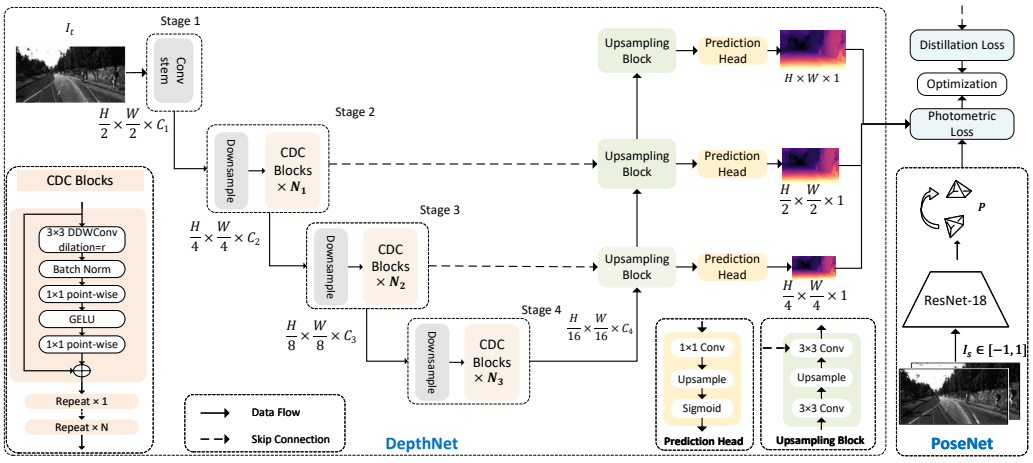

Figure 2. The training scheme of self-supervised depth estimation and the architectures of DepthNet and PoseNet.

3.2. SSDE Training Scheme

Figure 2 also shows the self-supervised depth estimation (SSDE) training scheme, which aims at minimizing the photometric loss $\mathcal{L}_p$ between a target image I_t and the

reconstructed one $\hat{I}_t$. The DepthNet takes a grayscale target image I_t and predicts a depth map D_t. The PoseNet estimates the relative pose P between the target image and an adjacent image $I_{t+s}, s \in [-1, 1]$.

3.2.1. Photometric Loss

With the adjacent image I_{t+s}, the estimated relative pose P, the predicted depth map D_t, and the camera's intrinsics K known, the target image can be reconstructed as a function $\mathcal{F}(I_{t+s}, P, D_t, K)$, and the photometric loss between the target image and the reconstructed image $\hat{I}_t$ can be defined as:

$$\mathcal{L}_p(\hat{I}_t, I_t) = \mathcal{L}_p(\mathcal{F}(I_{t+s}, P, D_t, K), I_t), \tag{1}$$

which can be calculated as a sum of the SSIM (structural similarity index) [51] and the L1 loss between two images:

$$\mathcal{L}_p(\hat{I}_t, I_t) = \alpha \frac{1 - SSIM(\hat{I}_t, I_t)}{2} + (1 - \alpha)\|\hat{I}_t - I_t\|, \tag{2}$$

with α being an empirical value of 0.85 [27]. The minimum photometric loss and the auto-masking techniques [27] are used to overcome the occlusion problem and to filter out objects that move at the same speed as the camera. The final photometric loss is defined as:

$$\mathcal{L}_p(\hat{I}_t, I_t) = \left\langle \min_{s\in[-1,1]} \mathcal{L}_p(I_{t+s}, I_t) > \min_{s\in[-1,1]} \mathcal{L}_p(\hat{I}_t, I_t) \right\rangle \cdot \min_{s\in[-1,1]} \mathcal{L}_p(\hat{I}_t, I_t), \tag{3}$$

where the $\langle\cdot\rangle$ operation outputs a binary mask.

3.2.2. Smoothness Loss

The edge-aware smoothness loss [27] is also used to improve the smoothness of the edges of the generated depth map:

$$\mathcal{L}_{smooth} = |\partial_x \mathrm{d}_t^*| e^{-|\partial_x I_t|} + |\partial_x \mathrm{d}_t^*| e^{-|\partial_y I_t|}, \tag{4}$$

where $d_t^* = d_t / \hat{d}_t$ is the mean-normalized inverse depth. Therefore, the combined loss function for the self-supervised training is:

$$\mathcal{L}_{ss} = \frac{1}{3} \sum_{j\in\{1,\frac{1}{2},\frac{1}{4}\}} (\mathcal{L}_p + \lambda \mathcal{L}_{smooth}), \tag{5}$$

where j can be three resolutions of the inverse depth. λ is set to 10^{-3} as in [27].

3.3. Knowledge Distillation Scheme

3.3.1. Matching Intermediate Features using the Channel Aware Distillation Transformer (CADiT)

Assume that the teacher network T and the student network S have intermediate feature maps denoted by $F_T \in \mathbb{F}^{H\times W\times C}$ and $F_S \in \mathbb{F}^{H\times W\times C'}$, respectively. The feature map has a height of H, a width of W, and channel numbers of C. As shown in Figure 3a, the student's feature channels are increased using 1×1 convolutions to have the same channel numbers as the teacher. Then, they can be reshaped as $F_T \in \mathbb{F}^{N\times C}$ and $F_S \in \mathbb{F}^{N\times C}$, respectively, where $N = H \times W$. The L2 loss can be used to minimize the discrepancy between the teacher's and student's intermediate features ($\mathcal{L}_{IF}$) [40]:

$$\mathcal{L}_{IF} = ||F_S - F_T||_2. \tag{6}$$

However, such a direct feature-matching method may increase the difficulty of optimization if the student has poor learning ability. The proposed CADiT (Figure 3b) allows

each student channel to learn geometric cues from all the teacher's channels to improve feature-matching. Specifically, a $C \times C$ channel correlation map (CCM) is built between the transposed aligned student and the teacher:

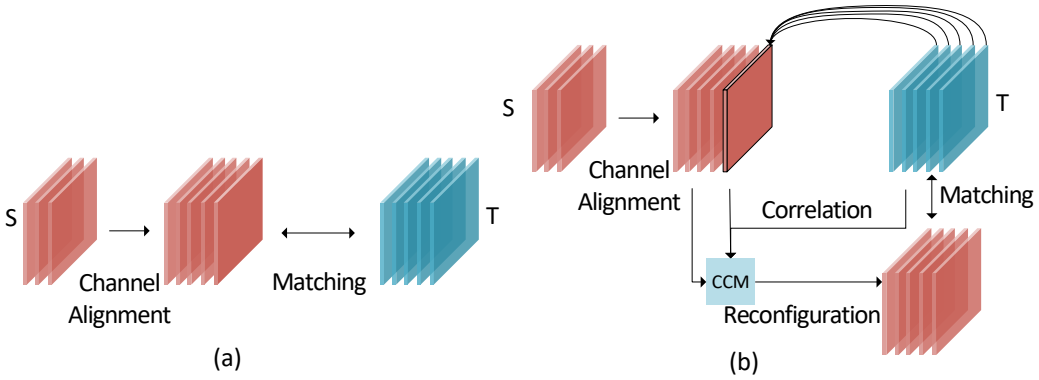

Figure 3. Intermediate feature-matching schemes. (**a**) is the conventional feature-matching scheme. (**b**) is the proposed CADiT that makes the student learn the channel correlations from the teacher.

$$CCM = Softmax(F_S^{\mathsf{T}} \cdot F_T) \in \mathbb{F}^{C \times C}, \tag{7}$$

where $(\cdot)$ is the inner product, and the CCM measures correlations between student and teacher channels. The student's features can be reconfigured as:

$$F_{S'} = F_S + F_S \cdot CCM \in \mathbb{F}^{N \times C}, \tag{8}$$

and the CADiT loss is computed as:

$$\mathcal{L}_{CADiT} = ||F_{S'} - F_T||_2. \tag{9}$$

3.3.2. Matching Outputs

As with the traditional KD methods [38,46,48], the L1 loss is also used to minimize the multi-scale depth maps generated by the teacher and the student. The $\mathcal{L}_{Out}$ is defined as:

$$\mathcal{L}_{Out} = \frac{1}{3} \sum_{j \in \{1, \frac{1}{2}, \frac{1}{4}\}} ||D_S - D_T||_1. \tag{10}$$

The final loss function to train the network, combining Equation (5), is written as:

$$\mathcal{L} = \mathcal{L}_{ss} + \alpha \mathcal{L}_{CADiT} + \mathcal{L}_{Out}, \tag{11}$$

where α is a weighting factor, set to 0.1 in this paper.

3.4. Drone Controlling

With the depth map generated by the network the nano-drone avoids obstacles based on the behavior arbitration (BA) scheme [52]. Although this scheme was originally used in conjunction with sonar, it can also be used with depth maps. The behavior *avoid* is defined as:

$$\dot{\phi} = f_{avoid}(\phi) = \lambda_{avoid} \sum_i [(\phi - \psi_i) \cdot e^{-c_{avoid} d_i} \cdot e^{-\frac{(\phi - \psi_i)^2}{2\sigma_i^2}})], \tag{12}$$

where λ_{avoid} is a weighting factor, ϕ is the heading of the drone, and d_i and ψ_i are the depth value and direction of the i-th value in the obstacle map, respectively. σ_i is the horizontal field of view of the camera, which is the angular range that the drone can change. The gain c_{avoid} controls the sensitivity to obstacles. Lower gain allows the drone to react to obstacles further away. Increasing λ_{avoid} changes the angular velocity of the drone more quickly. If

there is a single obstacle, the behavior will generate a repeller along the obstacle direction ψ_i. As distant obstacles have less importance than nearby ones, the repeller's magnitude should decrease as the distance from the obstacle increases.

The drone started to drift a little when it was about 1.5m above the ground because the drone's Kalman filter was not working stably. In this paper, the flight altitude of the drone is fixed at about 0.7 m, and an obstacle map is constructed by *average pooling* the center rows of the depth map with a window size of 10×10 (Figure 4), resulting in a 1D obstacle map. Although the method only uses the pixels in the horizontal center to construct the obstacle map, the complete images are needed to train the depth estimation network. It is not feasible to use only the center pixels during training, as there is no guarantee that the same pixels will be seen in the previous or next frame, and this violates the training using photometric loss. The considerations of using such a simple control strategy are as follows: (1) The calculation is cheap, and no additional trainable parameters are required to generate the obstacle map. (2) It allows the selection of the next waypoint based on the depth map for path planning in future work. It is possible to use more complex control schemes, but this is beyond the scope of this paper.

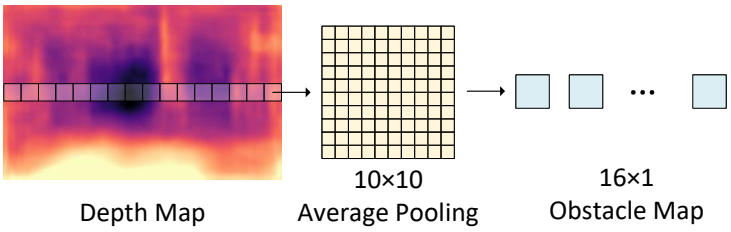

Figure 4. An obstacle map is generated by applying a sum pooling operation on the horizontal center of the depth map. The depth values of all the pixels in each pooling window are averaged.

In the implementation, the deep network and the controlling code do not run immediately when the drone is switched on. The drone needs about 12 s to initialize itself, connect to the laptop, and send several testing messages. A *take-off* command will then be given 15 s after the start. Five seconds later, the network starts, and then the controlling code runs. The drone will land either when a predefined flight time runs out or crashes.

4. Experiments

4.1. Drone Platform

To implement obstacle avoidance, this paper has considered several open-source drone platforms, including Crazepony, ArduBee, DJI Tello, and Crazyflie. A comparison of these platforms is listed in Table 1. Crazepony2 is a customizable platform, especially known for the First-Person View (FPV) drone. ArduBee is equipped with an optical flow sensor and an infrared sensor for object avoidance. Crazepony2 and ArduBee lack AI chips for on-board image processing. Although DJI Tello has a good vision processing unit and is designed for education, its closed-source system has become a hindrance to custom development. Bitcraze Crazyflie (Figure 5) is the drone platform used in this paper. It is equipped with a Flow Deck v2 at the bottom to measure the distance to the ground and an ultra-low power GAP8 processor (AI Deck) integrating a monochrome camera (HM01B0-MNA), which is designed for on-board AI computing. The active open-source community has also brought attention to Crazyflie. This drone platform measures $92 \times 92 \times 29$ mm $(W \times H \times D)$ and weighs 34 g. It is equipped with a 250 mAh LiPo battery and the maximum flight time is about 5 min. It has 22.65 GOPS of processing power and 512 KB of RAM.

Table 1. A comparison of the different open-source drone platforms.

Platform	Open-Source	MCU	Camera	AI Chip
Crazepony2	Yes	STM32F303	Yes	-
ArduBee	Yes	STM32H743VIT6	Yes	-
DJI Tello	No	Not Specified	Yes	Intel® Movidius™ VPUs
Crazyflie 2.1	Yes	STM32F405	Yes	GAP8

Figure 5. The Crazyflie is used as the drone platform in this paper.

4.2. Datasets

4.2.1. KITTI

KITTI is a multimodal dataset [53], which consists of 61 stereo road scenes. In this paper, the self-supervised model is trained on the Eigen split [54]. There are 39,810 monocular triplets used in the training, 4421 for evaluation, and 697 for testing. During training, all the RGB images in the KITTI dataset are resized to 192 × 640 and converted to one-channel grayscale images.

4.2.2. Gray Campus Indoor

This in-house indoor drone dataset is collected by a Crazyflie in different buildings on our campus. It consists of 17 sequences, a total of 9140 grayscale images with an original resolution of 244 × 324 pixels. Images are resized to 128 × 160 pixels in this paper to meet the requirement of running speed.

4.3. Implementation Details

4.3.1. Network Training

The proposed network is implemented in PyTorch. Models are trained for 30 epochs on KITTI and 100 epochs on Gray Campus Indoor with an NVIDIA TITAN Xp. The teacher model Lite-Mono is pretrained on ImageNet [55] and then trained on KITTI. During training, the teacher's weights are fixed, and only the student's weights are updated. AdamW [56] optimizer is used, and the weight decay is set to 10^{-4}. The initial learning rate is set to 10^{-4} with a cosine learning rate schedule.

4.3.2. Model Quantization and Deployment

The trained PyTorch model is further converted to the ONNX (Open Neural Network Exchange) format and quantized using an 8-bit scheme, reducing the size of the weights from 747.6 K bytes to 201.3 K bytes. The controlling algorithm is implemented in C language. The Fabric Controller (FC) frequency of the GAP8 is set to 250 MhZ.

4.4. Results on Grayscale KITTI

Table 2 shows the accuracy of models trained on the grayscale KITTI dataset, and the seven commonly used accuracy indicators [57] are AbsRel, SqRel, RMSE, RMSE log, $\delta < 1.25$, $\delta < 1.25^2$, and $\delta < 1.25^3$. By comparing the results of Lite-Mono with Lite-Mono (RGB) it can be found that the self-supervised training based on the photometric loss also works on grayscale images, albeit with less accuracy. This confirms the feasibility of the proposed method using grayscale images for self-supervised depth estimation. DDND

w/o KD in the table denotes the model that does not use knowledge distillation, i.e., it contains only the streamlined DepthNet and PoseNet as shown in Figure 2. The results show that leveraging the proposed KD method in the training the accuracy is greatly improved. Figure 6 shows some images generated by the networks, and it can be observed that DDND learns knowledge from Lite-Mono and is able to perceive larger objects. In addition, DDND can produce sharper depth maps at the edges of objects compared to the blurred depth maps produced without KD.

Table 2. Accuracy comparison on KITTI. "RGB" means that the training uses three-channel RGB images.

Method	Depth Error (↓)				Depth Accuracy (↑)			# Params.
	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	
Lite-Mono (RGB) [17]	0.107	0.765	4.561	0.183	0.886	0.963	0.983	3.1 M
Lite-Mono [17]	0.110	0.848	4.713	0.187	0.881	0.961	0.982	3.1 M
DDND w/o KD	0.157	1.259	5.593	0.229	0.796	0.930	0.973	0.31 M
DDND	0.147	1.149	5.394	0.221	0.813	0.936	0.974	0.31 M

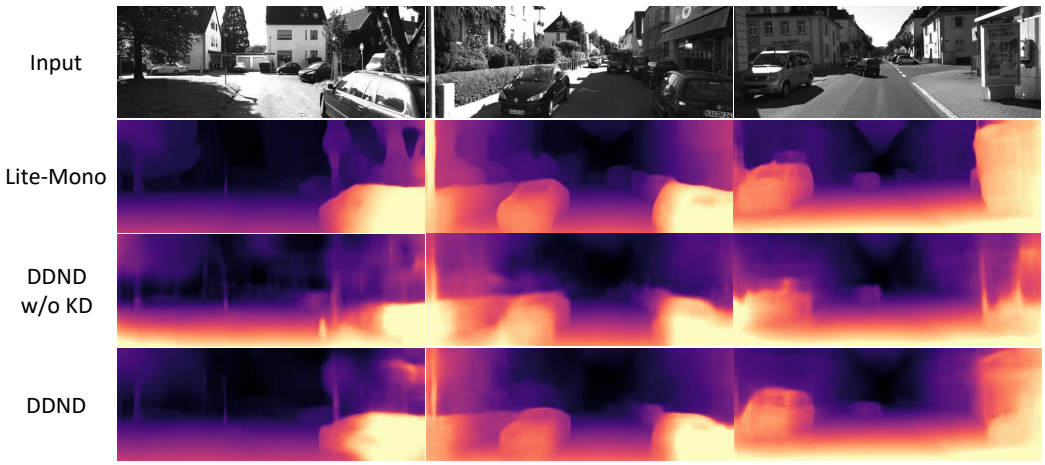

Figure 6. Qualitative results on KITTI. The student network can successfully learn feature representations from the teacher.

4.5. Qualitative Results on Gray Campus Indoor

Figure 7 shows some results on the in-house dataset. The dataset is challenging for the SSDE framework as it has many lighting sources and low-texture regions, such as walls and floors. In addition, scenes are more diverse. DDND benefits from the KD scheme and captures more detail in scenes.

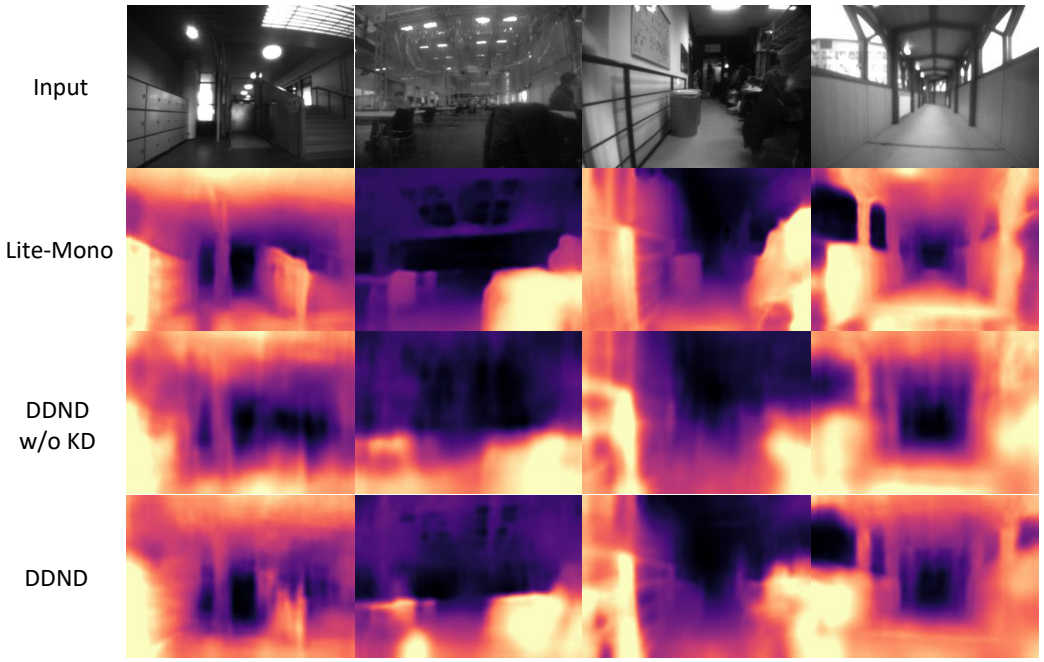

Figure 7. Qualitative results on Gray Campus Indoor.

4.6. Ablation Study on KD Losses

Ablation studies with different loss settings on KITTI are performed to validate the effectiveness of the proposed KD training and CADiT. In Table 3, the first setting is DDND without KD, which is the baseline in this ablation study. When introducing the KD on the generated depth maps (No. 2), the results are better than the baseline. However, experiment No. 3 shows that not all metrics are better when using the KD in the encoder at the same time. Simply using L2 loss to distill feature representations cannot give good results due to the large differences between student and teacher in network learning ability. From experiments No. 3 and No. 4, it can be found that the distillation methods using the channel loss yield better results. This loss allows the KD process to focus on the feature channels and enables more effective knowledge transfer. Experiments No. 6–8 show the effectiveness of the proposed CADiT module. Even if the proposed CADiT module is only used in the encoder, without the help of L1 loss in the decoder, good results can still be achieved (No. 7). The best result is obtained when the CADiT module is used in the encoder and the L1 distillation is used in the decoder.

Table 3. Accuracy comparison on KITTI of DDND using different KD losses. "E/D": KD in the encoder/decoder. "CD": channel loss proposed in [43]. The best results are highlighted in **bold**.

No.	KD Settings	Depth Error (↓)				Depth Accuracy (↑)		
		Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
1	E: n/a, D: n/a	0.157	1.259	5.593	0.229	0.796	0.930	0.973
2	E: n/a, D: L1	0.155	1.186	5.502	0.231	0.790	0.929	0.973
3	E: L2, D: L1	0.154	1.305	5.653	0.229	0.803	0.932	0.972
4	E: CD, D: n/a	0.155	1.242	5.649	0.228	0.797	0.930	0.973
5	E: CD, D: L1	0.152	1.208	5.561	0.226	0.807	0.934	0.973
6	E: n/a, D: CADiT+L1	0.149	1.172	5.472	0.226	0.807	0.933	**0.974**
7	E: CADiT, D: n/a	0.149	1.236	5.528	0.223	**0.815**	**0.936**	0.973
8	E: CADiT, D: L1	**0.147**	**1.149**	**5.394**	**0.221**	0.813	**0.936**	**0.974**

4.7. Test in Real Environments

The proposed approach is tested in real indoor environments. Figure 8 shows some images taken by the nano-drone, and the generated depth maps with the deployed quantized model. The green bars on the grayscale images denote steering commands for avoiding

obstacles calculated by Equation (12). Due to model quantization, the on-board network is not able to generate smooth depth maps, but these maps still succeed in showing the structure and volume of these scenes. Considering the inference speed of GAP8 the c_{avoid} defined in Equation (12) is set to 0.1 to make sure that the drone is able to react to obstacles at a safer distance.

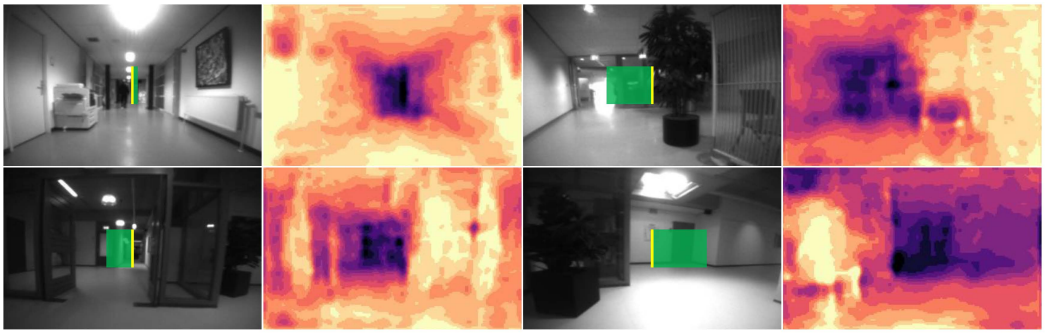

Figure 8. Real environment tests. Grayscale input images and their corresponding depth maps generated by the quantized CNN model are shown. The green bar in each grayscale image denotes the change in angular velocity to avoid obstacles.

4.8. Inference Speed Analysis

The inference speed of the proposed network is evaluated both on the NVIDIA TITAN Xp GPU (graphics processing unit) and the GAP8 processor. As shown in Table 4, there is little difference in the speed of the network inferring on TITAN Xp at either resolution, but on GAP8, it is about six times faster for the resolution of 128 × 160. It can also be observed that the computing power of edge computing devices such as GAP8 is extremely limited. The inference speed of 1.24 FPS is acceptable because the nano-drone flies at a low speed during tests.

Table 4. Inference speed evaluation under two image resolutions.

Resolution	**Speed (FPS)**	
	NVIDIA TITAN Xp	**GAP8**
128 × 160	434.78	1.24
224 × 320	431.53	0.22

4.9. Failure Cases

The proposed method fails to estimate the depth of the glass or if it is too close to a wall, as shown in Figure 9. This is also a limitation of SSDE methods, and this problem can be overcome by integrating additional sensors, such as ultrasonic sonar, to detect the distance between glass and walls.

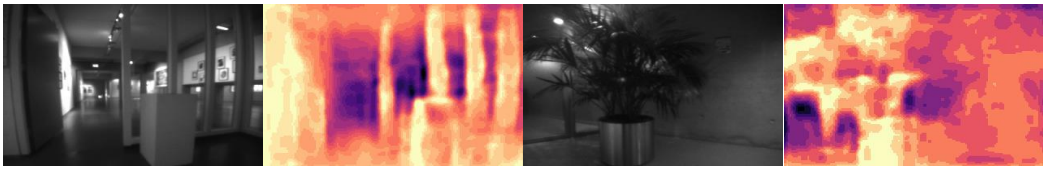

Figure 9. The method fails to avoid obstacles in areas of glass and when close to walls.

5. The Scene Reconstruction Pipeline

Since a map of the environment is useful for post-disaster rescue and it is not possible to run SLAM algorithms on the nano-drone, this paper also implements the streaming of collected images to a laptop using the nano-drone's WiFi module and presents an offline pipeline for reconstructing the environment. Figure 10 displays the entire pipeline, divided

into on-board processing and offline processing stages. During the on-board processing, the proposed depth estimation network runs on the nano-drone and generates relative depth maps and angular velocity to make the drone avoid obstacles.

Meanwhile, the grayscale images that have been captured are transmitted from the drone to a laptop through the NINA WiFi module of the GAP8 chip. On the laptop, a SLAM algorithm can be used to estimate the poses of the sequential images. This paper uses ORB-SLAM2 [57] to extract keyframe trajectories from the images. To create a 3D reconstruction, it is necessary to know the accurate depth values for each pixel, but the depth estimation models trained with a self-supervised scheme are only able to predict relative depth values. This paper adopts ZoeDepth [56] for metric depth estimation. ZoeDepth has 345 M parameters, and it has shown excellent generalization capacity as it was initially pretrained on 12 datasets using relative depth and subsequently fine-tuned on two datasets using metric depth. Then, the grayscale images and their corresponding depth maps are used to generate colored (monochrome) point clouds. Therefore, the scene reconstruction pipeline allows for building a map of an indoor environment by utilizing a nano-drone equipped with a monochrome camera.

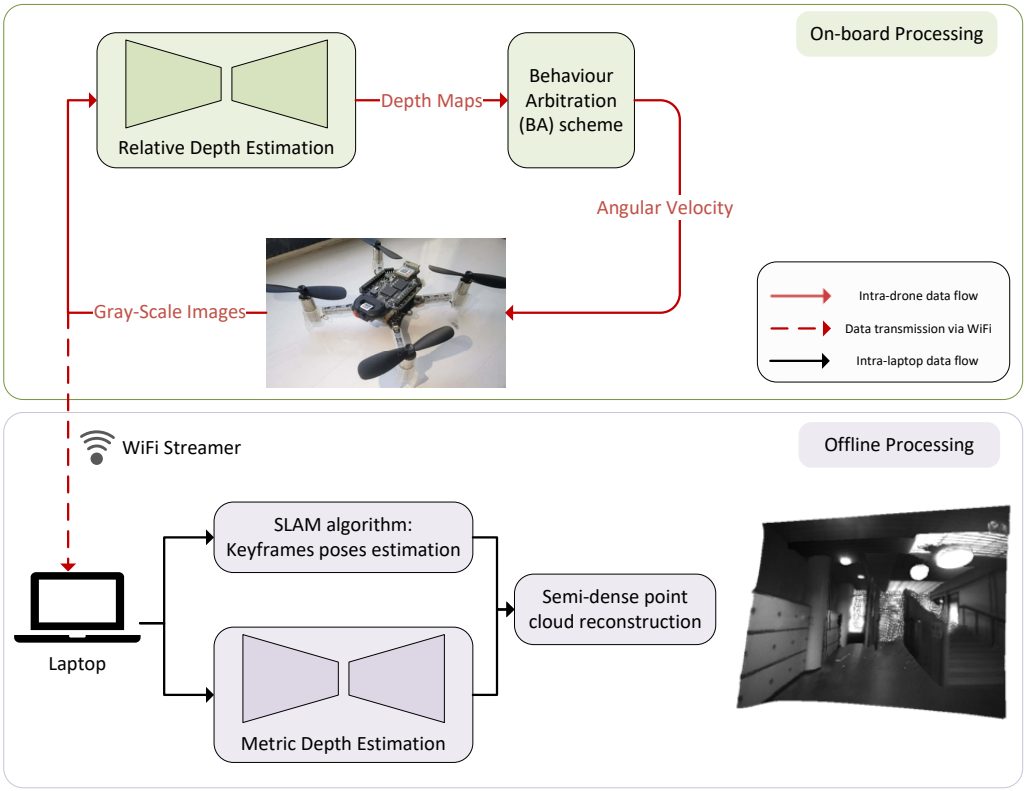

Figure 10. The application pipeline for the offline 3D reconstruction.

6. Conclusions

This paper proposes a lightweight depth estimation framework DDND, for obstacle avoidance on the nano-drone Crazyflie. Considering the limited storage and computing capacity of such a small drone platform, it is only possible to deploy a tiny network on it for monocular depth estimation. To enhance the learning ability of this tiny network, this paper integrates knowledge distillation and proposes the CADiT module for better knowledge transfer from a teacher model. The quantitative and qualitative results on the KITTI dataset validated the effectiveness of the proposed KD module. The model is then quantized so that it can infer on a Crazyflie for real environment tests. The limitation of such vision-based methods is that it is unable to avoid transparent objects such as glass. This paper also presents an application pipeline for the reconstruction of the environment using

offline metric depth estimation and keyframe pose estimation. With the 3D reconstruction, future potential work will be focused on the selection of waypoints in the reconstruction for path planning. This function requires the implementation of bilateral data communication between the laptop and the drone. The low inference speed of the algorithm only allows the drone to fly at a low speed. Future work will focus on improving the efficiency of the algorithm. A more powerful GAP9 chip is also being considered.

Author Contributions: Conceptualization, N.Z. and F.N.; methodology, N.Z.; software, N.Z.; validation, N.Z.; formal analysis, N.Z.; investigation, N.Z.; writing—original draft preparation, N.Z.; writing—review and editing, N.Z., F.N., G.V. and N.K.; visualization, N.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme and the Korean Government under Grant Agreement No. 833435. The content reflects only the authors' view and the Research Executive Agency (REA) and the European Commission are not responsible for any use that may be made of the information it contains.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Paliotta, C.; Ening, K.; Albrektsen, S.M. Micro indoor-drones (mins) for localization of first responders. In Proceedings of the ISCRAM, Blacksburg, VA, USA, 23–26 May 2021.
2. Smolyanskiy, N.; Gonzalez-Franco, M. Stereoscopic first person view system for drone navigation. *Front. Robot. AI* **2017**, *4*, 11. [CrossRef]
3. Schmid, K.; Tomic, T.; Ruess, F.; Hirschmüller, H.; Suppa, M. Stereo vision based indoor/outdoor navigation for flying robots. In Proceedings of the IROS, Tokyo, Japan, 3–8 November 2013; pp. 3955–3962.
4. Chiella, A.C.; Machado, H.N.; Teixeira, B.O.; Pereira, G.A. GNSS/LiDAR-based navigation of an aerial robot in sparse forests. *Sensors* **2019**, *19*, 4061. [CrossRef] [PubMed]
5. Moffatt, A.; Platt, E.; Mondragon, B.; Kwok, A.; Uryeu, D.; Bhandari, S. Obstacle detection and avoidance system for small uavs using a lidar. In Proceedings of the ICUAS, Athens, Greece, 1–4 November 2020; pp. 633–640.
6. Park, J.; Cho, N. Collision avoidance of hexacopter UAV based on LiDAR data in dynamic environment. *Remote Sens.* **2020**, *12*, 975. [CrossRef]
7. Akbari, A.; Chhabra, P.S.; Bhandari, U.; Bernardini, S. Intelligent exploration and autonomous navigation in confined spaces. In Proceedings of the IROS, Las Vegas, NV, USA, 25–29 October 2020; pp. 2157–2164.
8. Yang, T.; Li, P.; Zhang, H.; Li, J.; Li, Z. Monocular vision SLAM-based UAV autonomous landing in emergencies and unknown environments. *Electronics* **2018**, *7*, 73. [CrossRef]
9. von Stumberg, L.; Usenko, V.; Engel, J.; Stückler, J.; Cremers, D. From monocular SLAM to autonomous drone exploration. In Proceedings of the ECMR, Paris, France, 6–8 November 2017; pp. 1–8.
10. Tulldahl, M.; Holmberg, M.; Karlsson, O.; Rydell, J.; Bilock, E.; Axelsson, L.; Tolt, G.; Svedin, J. Laser sensing from small UAVs. In Proceedings of the Electro-Optical Remote Sensing XIV, San Francisco, CA, USA, 1–3 February 2020; Volume 11538, pp. 87–101.
11. Kouris, A.; Bouganis, C.S. Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation. In Proceedings of the IROS, Madrid, Spain, 1–5 October 2018.
12. Loquercio, A.; Maqueda, A.I.; Del-Blanco, C.R.; Scaramuzza, D. Dronet: Learning to fly by driving. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1088–1095. [CrossRef]
13. Gandhi, D.; Pinto, L.; Gupta, A. Learning to fly by crashing. In Proceedings of the IROS, Vancouver, BC, Canada, 24–28 November 2017; pp. 3948–3955.
14. Yang, X.; Chen, J.; Dang, Y.; Luo, H.; Tang, Y.; Liao, C.; Chen, P.; Cheng, K.T. Fast depth prediction and obstacle avoidance on a monocular drone using probabilistic convolutional neural network. *IEEE Trans. Intell. Transport. Syst.* **2019**, *22*, 156–167. [CrossRef]
15. Chakravarty, P.; Kelchtermans, K.; Roussel, T.; Wellens, S.; Tuytelaars, T.; Van Eycken, L. CNN-based single image obstacle avoidance on a quadrotor. In Proceedings of the ICRA, Singapore, 29 May–3 June 2017; pp. 6369–6374.
16. Zhang, Z.; Xiong, M.; Xiong, H. Monocular depth estimation for UAV obstacle avoidance. In Proceedings of the CCIOT, Changchun, China, 6–7 December 2019; pp. 43–47.
17. Zhang, N.; Nex, F.; Vosselman, G.; Kerle, N. Lite-mono: A lightweight cnn and transformer architecture for self-supervised monocular depth estimation. In Proceedings of the CVPR, Vancouver, BC, Canada, 18–22 June 2023; pp. 18537–18546.
18. McGuire, K.; De Wagter, C.; Tuyls, K.; Kappen, H.; de Croon, G.C. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Sci. Robot.* **2019**, *4*, eaaw9710. [CrossRef]

19. Duisterhof, B.P.; Li, S.; Burgués, J.; Reddi, V.J.; de Croon, G.C. Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In Proceedings of the IROS, Prague, Czech Republic, 27 September–1 October 2021; pp. 9099–9106.
20. Niculescu, V.; Müller, H.; Ostovar, I.; Polonelli, T.; Magno, M.; Benini, L. Towards a Multi-Pixel Time-of-Flight Indoor Navigation System for Nano-Drone Applications. In Proceedings of the I2MTC, Ottawa, ON, Canada, 16–19 May 2022; pp. 1–6.
21. Vanhie-Van Gerwen, J.; Geebelen, K.; Wan, J.; Joseph, W.; Hoebeke, J.; De Poorter, E. Indoor drone positioning: Accuracy and cost trade-off for sensor fusion. *IEEE Trans. Veh. Technol.* **2021**, *71*, 961–974. [CrossRef]
22. Briod, A.; Zufferey, J.C.; Floreano, D. Optic-flow based control of a 46g quadrotor. In Proceedings of the IROS Workshop, Tokyo, Japan, 3–8 November 2013.
23. Bouwmeester, R.J.; Paredes-Vallés, F.; de Croon, G.C. NanoFlowNet: Real-time Dense Optical Flow on a Nano Quadcopter. *arXiv* **2022**, arXiv:2209.06918.
24. McGuire, K.; De Croon, G.; De Wagter, C.; Tuyls, K.; Kappen, H. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1070–1076. [CrossRef]
25. Palossi, D.; Loquercio, A.; Conti, F.; Flamand, E.; Scaramuzza, D.; Benini, L. A 64-mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet Things J.* **2019**, *6*, 8357–8371. [CrossRef]
26. Zhilenkov, A.A.; Epifantsev, I.R. System of autonomous navigation of the drone in difficult conditions of the forest trails. In Proceedings of the EIConRus, Moscow, Russia, 29 January–1 February 2018; pp. 1036–1039.
27. Godard, C.; Mac Aodha, O.; Firman, M.; Brostow, G.J. Digging into self-supervised monocular depth estimation. In Proceedings of the ICCV, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3828–3838.
28. Jung, H.; Park, E.; Yoo, S. Fine-grained semantics-aware representation enhancement for self-supervised monocular depth estimation. In Proceedings of the ICCV, Montreal, QC, Canada, 11–17 October 2021; pp. 12642–12652.
29. Yin, Z.; Shi, J. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In Proceedings of the CVPR, Salt Lake City, UT, USA, 18–22 June 2018; pp. 1983–1992.
30. Poggi, M.; Aleotti, F.; Tosi, F.; Mattoccia, S. On the uncertainty of self-supervised monocular depth estimation. In Proceedings of the CVPR, Seattle, WA, USA, 16–18 June 2020; pp. 3227–3237.
31. Yang, N.; Stumberg, L.v.; Wang, R.; Cremers, D. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In Proceedings of the CVPR, Seattle, WA, USA, 16–18 June 2020; pp. 1281–1292.
32. Yan, J.; Zhao, H.; Bu, P.; Jin, Y. Channel-wise attention-based network for self-supervised monocular depth estimation. In Proceedings of the 3DV, Online, 1–3 December 2021; pp. 464–473.
33. Bae, J.; Moon, S.; Im, S. Deep Digging into the Generalization of Self-supervised Monocular Depth Estimation. In Proceedings of the AAAI, Washington, DC, USA, 7–14 February 2023.
34. Lyu, X.; Liu, L.; Wang, M.; Kong, X.; Liu, L.; Liu, Y.; Chen, X.; Yuan, Y. Hr-depth: High resolution self-supervised monocular depth estimation. In Proceedings of the AAAI, Online, 2–9 February 2021; Volume 35, pp. 2294–2301.
35. Wofk, D.; Ma, F.; Yang, T.J.; Karaman, S.; Sze, V. Fastdepth: Fast monocular depth estimation on embedded systems. In Proceedings of the ICRA, Montreal, QC, Canada, 20–24 May 2019; pp. 6101–6108.
36. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
37. Zhou, Z.; Fan, X.; Shi, P.; Xin, Y. R-msfm: Recurrent multi-scale feature modulation for monocular depth estimating. In Proceedings of the ICCV, Montreal, QC, Canada, 11–17 October 2021; pp. 12777–12786.
38. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
39. Zhang, Y.; Xiang, T.; Hospedales, T.M.; Lu, H. Deep mutual learning. In Proceedings of the CVPR, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4320–4328.
40. Romero, A.; Ballas, N.; Kahou, S.E.; Chassang, A.; Gatta, C.; Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv* **2014**, arXiv:1412.6550.
41. Komodakis, N.; Zagoruyko, S. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In Proceedings of the ICLR, Toulon, France, 24–26 April 2017.
42. Shu, C.; Liu, Y.; Gao, J.; Yan, Z.; Shen, C. Channel-wise knowledge distillation for dense prediction. In Proceedings of the ICCV, Montreal, QC, Canada, 11–17 October 2021; pp. 5311–5320.
43. Zhou, Z.; Zhuge, C.; Guan, X.; Liu, W. Channel distillation: Channel-wise attention for knowledge distillation. *arXiv* **2020**, arXiv:2006.01683.
44. Wang, Y.; Li, X.; Shi, M.; Xian, K.; Cao, Z. Knowledge distillation for fast and accurate monocular depth estimation on mobile devices. In Proceedings of the CVPR, Nashville, TN, USA, 19–25 June 2021; pp. 2457–2465.
45. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the CVPR, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
46. Hu, J.; Fan, C.; Jiang, H.; Guo, X.; Gao, Y.; Lu, X.; Lam, T.L. Boosting Light-Weight Depth Estimation Via Knowledge Distillation. *arXiv* **2021**, arXiv:2105.06143.
47. Pilzer, A.; Lathuiliere, S.; Sebe, N.; Ricci, E. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In Proceedings of the CVPR, Long Beach, CA, USA, 16–20 June 2019; pp. 9768–9777.

48. Cho, J.H.; Hariharan, B. On the efficacy of knowledge distillation. In Proceedings of the ICCV, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 4794–4802.
49. Stanton, S.; Izmailov, P.; Kirichenko, P.; Alemi, A.A.; Wilson, A.G. Does knowledge distillation really work? *NeurIPS* **2021**, *34*, 6906–6919.
50. Lin, S.; Xie, H.; Wang, B.; Yu, K.; Chang, X.; Liang, X.; Wang, G. Knowledge distillation via the target-aware transformer. In Proceedings of the CVPR, New Orleans, LA, USA, 21–23 June 2022; pp. 10915–10924.
51. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
52. Althaus, P.; Christensen, H.I. Behaviour coordination for navigation in office environments. In Proceedings of the IROS, Lausanne, Switzerland, 30 September–4 October 2002; Volume 3, pp. 2298–2304.
53. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Res.* **2013**, *32*, 1231–1237. [CrossRef]
54. Eigen, D.; Fergus, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In Proceedings of the ICCV, Santiago, Chile, 13–16 December 2015; pp. 2650–2658.
55. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the CVPR, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
56. Bhat, S.F.; Birkl, R.; Wofk, D.; Wonka, P.; Müller, M. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv* **2023**, arXiv:2302.12288.
57. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]