Nano Quadcopter Obstacle Avoidance with a Lightweight Monocular Depth Network

Liu, Cheng; Xu, Yingfu; van Kampen, Erik Jan; de Croon, Guido

DOI
10.1016/j.ifacol.2023.10.217

Publication date
2023

Document Version
Final published version

Published in
IFAC-PapersOnLine

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-2 (2023) 9312–9317

Nano Quadcopter Obstacle Avoidance with a Lightweight Monocular Depth Network

Cheng Liu * **Yingfu Xu** * **Erik-Jan van Kampen**
Guido de Croon

* *Equal contribution, in alphabetical order.*
Aerospace Engineering, Delft University of Technology (e-mail: {c.liu-10, y.xu-6, e.vankampen, g.c.h.e.decroon}@tudelft.nl)

Abstract:
In this paper, we propose an obstacle avoidance solution for a 34-gram quadcopter equipped with a monocular camera. The perception of obstacles is tackled by a lightweight convolutional neural network predicting a dense depth map from a captured grey-scale image. The depth network performs self-supervised learning and thus requires no ground-truth labels that are costly to acquire. Based on the depth map, the control strategy is implemented by a behavior state machine that balances the efficiency to explore the environment and the safety of avoiding obstacles. In real-world flight experiments, our solution demonstrates the efficacy of predicting trust-worthy depth maps and a stable control strategy in various cluttered environments.

Keywords: Autonomous robotic systems, Perception and sensing, Flying robots, Embedded robotics, Monocular depth prediction, Self-supervised learning.

1. INTRODUCTION

Small flying robots, such as nano quadcopters, have promising applications owning to their agility in narrow spaces. However, the small size and light weight (*e.g.*, ∼ 10×10 cm and ∼30 grams) limit the sensing and processing resources onboard. A monocular camera is small in size and can sense rich information, making it the primary sensor that enables a flying robot to interact with its surroundings. Besides, to navigate a nano quadcopter autonomously and safely, the visual processor and controller are required to be efficient enough to ensure low latency. This is especially important in cluttered environments where obstacle avoidance is a necessity.

Focusing on learning-based solutions, there are various ways to navigate a quadcopter between obstacles with a monocular camera. The first is using an end-to-end navigation network. The network proposed in Loquercio et al. (2018) was trained by human driving actions and can directly generate a obstacle-avoiding navigation policy. The disadvantage of end-to-end networks is that it is hard to reasoning the strategy they follow, which makes it harder to predict the strategy's capability of generalizing to unknown environments.

The second way is to infer the depth of a pixel from its optical flow, assuming the surroundings to be stationary. The authors of Bouwmeester et al. (2022) focused on substantially reducing neural network size while retaining sufficient performance for dense optic flow estimation. The network was applied to obstacle avoidance by comparing the left and right halves of the flow map. The disadvantage of using optic flow for obstacle avoidance is that sufficient motion is essential and that obstacles in the direction of motion are hard to detect.

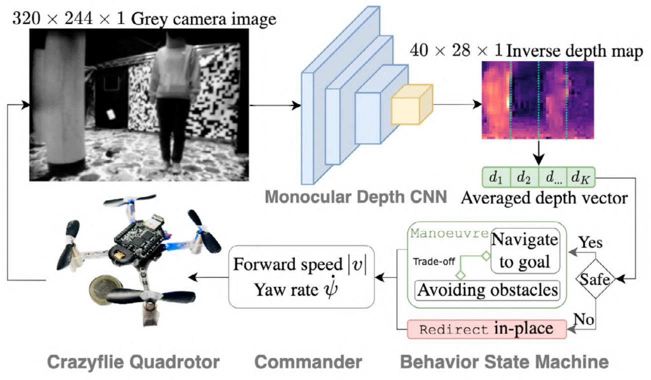


Fig. 1. System architecture. In our framework, a lightweight CNN is proposed for monocular depth estimation on a nano quadcopter, e.g. Crazyflie (Bitcraze, 2022). Based on the depth estimation, a behavior state machine is used to `manoeuvre` the quadcopter around obstacles while exploring the environment, and to `redirect` it to safe directions when facing dangerous scenarios.

The third way is resorting to a dense depth map. It is popular with flying robots equipped with a stereo or RGB-D (depth) camera. Thanks to recent developments in monocular depth estimation with deep neural networks (DNNs), a dense depth map can be obtained from a single image. There are not only works pursuing higher accuracy with larger networks Godard et al. (2019); Ranftl et al. (2020) but also works focusing on smaller networks and faster inference speed, with the aim of deployment on *memory-and-computation-constraint* mobile devices. Many of them adopted self-supervised learning Liu et al. (2020); Poggi et al. (2022). To better train a small network, learning

10.1016/j.ifacol.2023.10.217

from a larger network through knowledge distillation is also a popular choice Aleotti et al. (2020); Yucel et al. (2021); Wang et al. (2021). Dense depth maps are applied to obstacle avoidance of a quadcopter in Chakravarty et al. (2017). A disadvantage is that the depth network requires ground-truth labels for training. Additionally, the control strategy is not capable of handling corner cases, such as being too close to a wall or obstacles in the center view.

In this work, we develop an obstacle avoidance framework (as shown in Fig. 1) for a nano quadcopter based on a lightweight depth network (LDN). The contributions can be summarized as:

(1) The training of the LDN combines two supervision signals: self-supervised learning using reprojection-based view synthesis and knowledge distillation from a larger network. Better accuracy is achieved than using either one of them solely. No ground-truth label is required.
(2) The behavior state machine handles noisy low-resolution depth maps and safely navigates the quadcopter out of corner cases, such as dynamic obstacles, obstacles right in front, or facing a planar wall vertically. This approach improves upon Chakravarty et al. (2017); Bouwmeester et al. (2022) in generating robuster control commands.

The structure of the paper is as follows. Section 2 describes the architecture of the LDN, learning schemes, datasets, and the design of a behavior state machine. In Section 3, we present an ablation study of learning schemes and the results of flight experiments of a nano quadcopter. The conclusion is followed in Section 4.

2. METHODOLOGY

2.1 Lightweight Monocular Depth Network

The architecture of the LDN is shown in Fig. 2. Most depth networks in the literature have the downsampling encoder and upsampling decoder architecture, predicting a depth map with the same resolution as the input image. Differently, our network performs downsampling three times and reaches 1/8th resolution of the input image. There is upsampling operation. It has 391, 793 trainable parameters in total. According to our knowledge of literature, only the MiniNet Liu et al. (2020) has fewer parameters than the proposed network. MiniNet has an iterative recurrent module, while our network performs a one-time forward pass. Note that we do not compare the proposed LDN with other works because our focus is on ground-truth-free training schemes instead of network architecture design. The training schemes studied can be applied to a depth network with any architecture.

The reason for predicting low-resolution depth maps is two folds. Firstly, the network aims to run onboard a Crazyflie nano quadcopter equipped with a Bitcraze AI-deck[1] DNN processor. A PyTorch-trained network model is required to be converted by specific software to be flashed into the AI-deck. The proposed network architecture is empirically optimized under the given hardware/software constraints. Secondly, our control strategy does not require a high-resolution depth map. Given the resolution of the image captured by the onboard camera (324×244), we crop it to 320×224, which means the 1/8th resolution depth map has 40×28 pixels. The resolution may seem low, but it provides adequate information about the surroundings to fly safely. It is also sufficient for self-supervised learning and knowledge distillation, which will be introduced later.

[1] `https://www.bitcraze.io/products/ai-deck/`

2.2 Depth Learning Schemes

We discuss two supervision signals for training an LDN. The first is the reprojection-based self-supervised learning. The main loss function is the photometric matching loss $L_{\text{photo}}(D,T)$. D denotes the depth map prediction of image I_i and T is the relative pose prediction between I_i and I_{i+1}. We adopt Monodepth2 Godard et al. (2019) as the base and add the geometry consistency loss L_{geo} proposed in Bian et al. (2019). Since the LDN outputs low-resolution depth maps, the grey-scale images constructing L_{photo} are downsampled to 1/8th resolution. T can be predicted by a pose network trained from scratch simultaneously with LDN. The training of LDN-1 in Table 1 uses this scheme.

T can be obtained in another way. Given the mechanism of $L_{\text{photo}}(D,T)$, more accurate pose predictions can benefit the training of an LDN. Following this idea, we train the depth and pose networks of Monodepth2 that are based on ResNet-18 with full-resolution images. Besides adding the geometry consistency loss, to improve the pose accuracy, we implement the iterative depth-dependent pose prediction Wagstaff et al. (2022). The number of iterations is set to three in both training and inference. When loading image snippets during training, we implement that there is a 75% possibility of using a bigger temporal step. Because the relative translational motion between I_i and I_{i+2} is usually bigger than I_i and I_{i+1}, which is better for learning depth. The trained Monodepth2 networks are referred to as *Teacher* networks in the rest of this article. The pose predictions from *Teacher* networks are used in the reprojection-based loss, as in the case of training LDN-3, LDN-4, and LDN-5 in Table 1. *Teacher* networks perform only inference in the training of an LDN.

The second supervision signal is the knowledge distillation (KD) loss from the depth *Teacher* network. A predicted low-resolution depth map D_{LDN} is upsampled via bilinear interpolation to be the same resolution as the input image. The depth map prediction of *Teacher* network D_T serves as the proxy label. We use the $L1$ loss, $L_{\text{KD}} = |D_T - up(D_{\text{LDN}})|$. The KD loss weight $\lambda_{\text{KD}} = 1.0$. The complete loss function is shown in Eq. (1).

$$L = \lambda_{\text{geo}} \cdot L_{\text{geo}}(D_{\text{LDN}},T) + L_{\text{photo}}(D_{\text{LDN}},T) + \lambda_{\text{KD}} \cdot L_{\text{KD}} \tag{1}$$

2.3 Image Datasets

We collect two datasets in two real-world environments. The grey-scale images captured by the tiny camera onboard the quadcopter were transmitted via wireless communication to a laptop computer. During data collection, we held the quadcopter in hand and performed random motion. One environment is a controllable experimental environment *CyberZoo* (Fig. 3 (a)). In this environment the layout of objects is easy to change in order to test with different conditions, *e.g.*, density and type of obstacles. 6,231 images were captured in total. Another environment

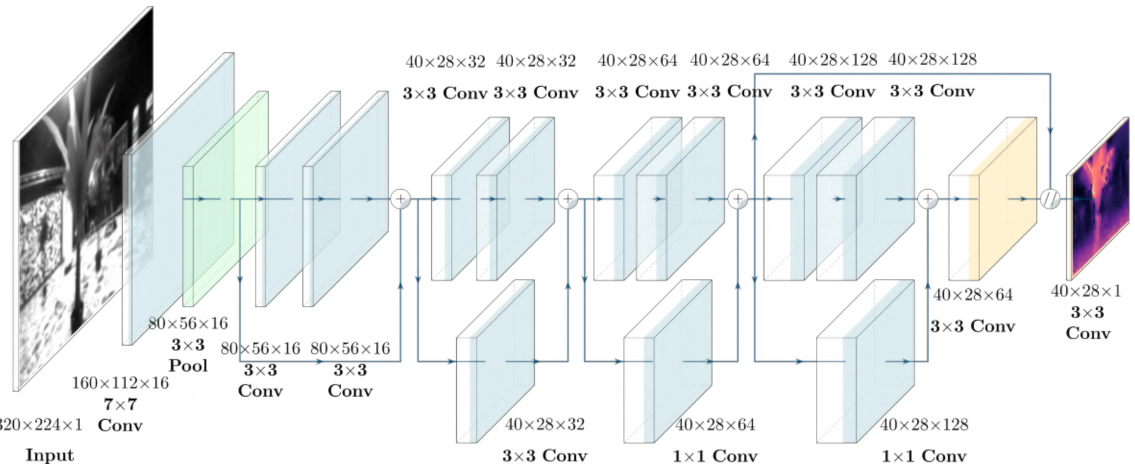
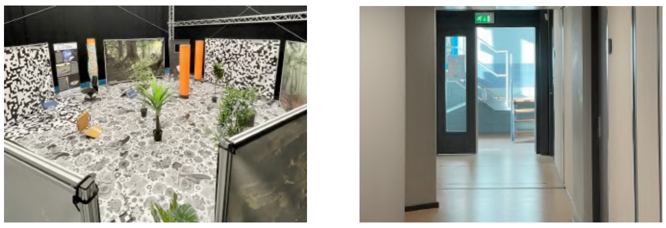


Fig. 2. Architecture of the lightweight CNN for monocular depth prediction. "+" denotes element-wise summation and "//" stands for tensor concatenation along the channel dimension. Blue color denotes that the convolutional layer is followed by a batch normalization layer and a ReLU activation function. Color yellow denotes that the convolutional layer is followed by a ReLU6 activation function. Color green denotes a max pooling layer with stride=2.

is a *Corridor* in an office building (Fig. 3 (b)). Here we captured 2,717 images. There is no ground-truth label available. All the images are used for training.



(a) *CyberZoo* environment. The obstacles include screens, chairs, plants, pillars, *etc.*

(b) *Corridor* that can contain trashcans and chairs.

Fig. 3. Real-world environments for data collection.

Besides the real-world datasets, we take three subsets from the TartanAir Wang et al. (2020) dataset to evaluate the training schemes, as shown in Table 1. Raw images and ground-truth depth maps are resized to the resolution of 320×224. The subset of the top group of Table 1 is made of *office* and *office2* environments. The second subset is collected in the *neighborhood* environment. The subset of the group at the bottom of the table contains the *seasonsforest* and *seasonsforest winter* environments. Around 15% images are used for testing. In accuracy evaluation, an LDN is trained on a single dataset and tested on the same dataset.

2.4 Behavior State Machine for High-Level Control

We design an inverse-depth-map-based Behavior State Machine (BSM) to generate high-level velocity commands for the quadcopter. The velocity command is then tracked by a low-level attitude controller. By saying inverse, we mean that nearer obstacles have larger inverse depth values which are restricted to the range $d \in (0, 1)$. As shown in Fig. 1, rather than focusing on pixel-wise depth values, we firstly extract a K-dimensional vector by taking the average of K sub-maps cut along the column of the full map, where d_k is the k-th averaged inverse depth value. Our motivation for doing this is to increase robustness to trivial depth changes restricted to small areas. Furthermore, it is expected that in this way the state machine can be also insensitive to depth prediction errors and image noise.

As shown in Fig. 1, one state of the BSM is `manoeuvre` which implements the trade-off between navigation to a goal and avoiding surrounding obstacles. The obstacle avoidance control is mainly inspired by the Behavior Arbitration Scheme Althaus and Christensen (2002); Chakravarty et al. (2017), in which all elements in the averaged depth vector are weighted summed to generate the yaw rate command $\dot{\psi}$:

$$\dot{\psi}_{\text{avoid}}(d) = \lambda_{avoid} \sum_{k=0}^{K-1} [(\frac{K+1}{2} - k) \cdot e^{-\frac{c}{d_k}} \cdot e^{-\frac{(k-\frac{K+1}{2})^2}{2\sigma^2}}] \quad (2)$$

where $\lambda_{\text{avoid}}, c, \sigma$ are weights that can be tuned for different sensitivity levels to obstacles. In principle, obstacles with higher inverse depth values that are near to the center view of the quadcopter contribute to a larger $\dot{\psi}$.

During `manoeuvre`, the forward speed is set to be $|v| = 0.3\text{m/s}$. Besides, to encourage the quadcopter to explore the environment rather than focusing on avoiding obstacles in a local area, the goal direction of navigation is chosen to be the vector index that corresponds to the smallest inverse depth value:

$$\dot{\psi}_{\text{goal}}(d) = \lambda_{goal}(\arg\min_{k}(d_k) - \frac{K+1}{2}) \quad (3)$$

The overall yaw rate can be summarized as:

$$\dot{\psi} = \dot{\psi}_{\text{avoid}}(d) + \dot{\psi}_{\text{goal}}(d). \quad (4)$$

The other state of BSM is `redirect`, redirecting the facing of the quadcopter to a safer direction at a certain yaw rate while hovering. This state is triggered when either of these two situations is satisfied: (1) when the difference Δ between the mean of the two elements in the center of the averaged depth vector and the mean of other elements is larger than threshold η; (2) when the variance of the K vector values is below a threshold ϵ. The first case represents an obstacle in the center view, on the other hand, case (2) indicates that there maybe a planar wall right in front. By tuning the weight parameters and thresholds, the behavior state machine can achieve decent exploration performance while avoiding obstacles and escape from corner cases. Specifically, we manually tuned $K = 8$ in the flight experiments. Algorithm 1 describes the proposed BSM in detail.

Algorithm 1: Behavior State Machine (BSM)

```
Inputs: $K, \lambda_{avoid}, \lambda_{goal}, c, \sigma, \eta, \epsilon$; trained params $\theta$
Initialize monocular depth CNN with params $\theta$
while true do
    Get inverse depth map through CNN inference
    Get the averaged depth vector $\mathbf{D} = [d_1, d_2, ..., d_K]$
    Calculate center indexes $k_1 = \frac{K}{2}, k_2 = \frac{K}{2} + 1$
    Calculate depth value difference
      $\Delta = \frac{1}{2}(d_{k_1} + d_{k_2}) - \frac{1}{K-2}\sum_k d_k, k \neq k_1, k_2$
    if $\Delta > \eta$ or $Var[\mathbf{D}] < \epsilon$ then
        redirect while hovering in-place
    else
        Calculate yaw rate command (2), (3), (4)
        manoeuvre
end
```

3. EVALUATION

In this section, we first compare the accuracy of LDNs trained by different learning schemes using the three public datasets whose ground-truth depth maps are available. Secondly, We conduct experiments in cluttered environments and demonstrate the effectiveness of the LDNs and the behavior state machine. Finally, we test the generalization capability of an LDN.

3.1 Comparison of Learning Schemes

To explore the best way to train an LDN, we compare five training schemes by training LDNs on three datasets, corresponding to the three groups divided by horizontal lines in Table 1. The "Teacher" in the first row of each group denotes *Teacher* networks trained on the individual dataset. They are the pre-trained teacher (PTT) networks in the training of LDN-2, LDN-3, and LDN-4. The only difference between LDN-5 and LDN-4 is that *Teacher* networks are trained from scratch simultaneously with LDN-5. The gradient flows of *Teacher* networks' outputs are detached from the loss function for LDN training. So the training of the *Teacher* networks of LDN-5 is not affected by LDN-5.

Ground-truth depth maps are downsampled to the same resolution as LDN predictions for accuracy evaluation. Many images have pixels filming the texture-less sky, whose correct depth is hard to be learned by the reprojection-based loss. Therefore we exclude from the evaluation all pixels whose ground-truth depth is more than 100 meters. We follow the same evaluation procedure as Godard et al. (2019). Seven metrics of depth prediction accuracy shown in the first row of Table 1 are calculated after performing per-image median ground truth scaling. The first four count the error of all pixels. $\delta < 1.25^n$ indicate the ratio of the predictions whose differences to the ground-truth values lay within the thresholds, *i.e.*, close enough to the ground truth.

From the data in Table 1, we notice that using KD as the only supervision signal (LDN-2) may lead to outlier depth predictions that are very inaccurate, as indicated by the big *Sq Rel* values in the first and the third groups. Using both KD and SSL but without the geometry consistency loss (LDN-3) leads to outliers as well. Using SSL loss alone to train the LDN and a pose network from scratch (LDN-1) produces few apparent outliers. But comparing LDN-1's accuracy with LDN-4, the gap is evident, especially in the metrics $\delta < 1.25^n$. Combining all loss terms, LDN-4 has the overall best performance. In all metrics, the accuracy of LDN-5 is only slightly worse than LDN-4. An advantage of LDN-5 over LDN-4 is that LDN-5 requires only one training process. Training the LDN does not require waiting for the completion of training *Teacher* networks, which reduces the time required for deployment in a new target environment.

3.2 Flight Experiments

We choose the training scheme of the LDN-5 of Table 1 for the networks that navigate the quadcopter. Two LDNs are trained respectively on the datasets collected in the two real-world environments. To better evaluate the performance of the LDN and improve the control strategy, all subsequent experiments are conducted with off-board processing. The grey-scale images captured by the quadcopter's camera are wirelessly transmitted to a laptop computer for inference, where high-level control commands are generated. The inference frequency, restricted mainly by image transfer, is around 7 Hz.

CyberZoo Experiments Four different environments were tested in CyberZoo, featuring sparse static obstacles, dense static obstacles, dynamic obstacles, and unknown obstacles. The quadcopter successfully navigates through environments containing sparse and dense static obstacles without collisions, covering a wide area and avoiding obstacles, as depicted in Fig. 4. Videos of experiments in environments with dynamic and unknown obstacles are available in the supplementary material[2].

To study how exactly the depth network and the BSM works, we investigate the *green* trajectory in Fig. 4 (b). The high-level control command at each time-step along this trajectory and the corresponding behavior state are shown in Fig. 5. It can be seen that the BSM is able to consistently switch between `manoeuvre` and `redirect` at different scenarios and generate yaw rate commands to navigate through the cluttered environment. In the left column of Fig. 6, we show the camera views and depth maps at timesteps $t_1 \sim t_5$. The monocular depth network's predictions of the inverse depth map provide crucial information for the BSM to detect and avoid obstacles in the environment.

At t_1 and t_5, the quadcopter is in `manoeuvre` mode due to the close proximity to obstacles, as shown in left column

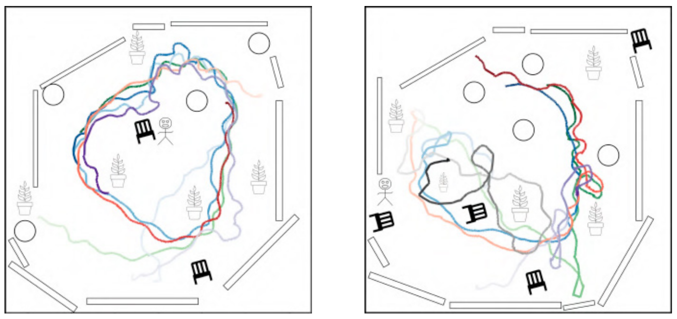
[2] `https://github.com/tudelft/depth_avoider_crazyflie.git`

Table 1. Ablation Study. **Bold** numbers indicate the best accuracy metrics achieved by LDNs.

Model	SSL[1]	λ_{geo}	KD[2]	PTT	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	$\delta < 1.25$↑	$\delta < 1.25^2$↑	$\delta < 1.25^3$↑
Teacher	✓	0.2			0.0925	0.8052	4.7785	0.2378	91.30 %	95.75 %	97.20 %
LDN-1	✓	0.02			0.2735	1.3616	5.9309	0.4399	57.38 %	82.78 %	91.74 %
LDN-2			✓	✓	0.2542	260.60	22.072	0.3115	**83.76** %	**93.67** %	**96.26** %
LDN-3	✓	0.0	✓	✓	0.2537	397.25	21.415	0.3124	83.30 %	93.64 %	96.24 %
LDN-4	✓	0.2	✓	✓	**0.1574**	**1.0726**	**5.4606**	**0.3104**	82.80 %	93.30 %	96.10 %
LDN-5	✓	0.2	✓		0.1617	1.1024	5.4925	0.3165	81.77 %	92.99 %	95.96 %
Teacher	✓	0.2			0.2373	2.1778	7.1647	0.3184	71.15 %	86.23 %	92.57 %
LDN-1	✓	0.02			0.3888	4.2736	10.552	0.5167	45.72 %	70.60 %	82.73 %
LDN-2			✓	✓	0.3202	3.7768	**8.2885**	0.3936	57.87 %	80.12 %	89.70 %
LDN-3	✓	0.0	✓	✓	**0.3186**	7.7472	8.4581	**0.3925**	**57.91** %	**80.20** %	**89.80** %
LDN-4	✓	0.2	✓	✓	0.3204	**2.8655**	8.5541	0.4044	56.88 %	79.00 %	88.87 %
LDN-5	✓	0.2	✓		0.3232	2.9012	8.5883	0.4064	56.58 %	78.76 %	88.76 %
Teacher	✓	0.2			0.4596	5.1482	10.596	0.5282	58.06 %	76.01 %	84.18 %
LDN-1	✓	0.02			**0.5374**	7.7569	14.000	0.6515	47.32 %	65.97 %	76.04 %
LDN-2			✓	✓	0.5945	48.237	13.878	0.5753	52.91 %	72.29 %	81.61 %
LDN-3	✓	0.0	✓	✓	0.5659	33.697	16.351	**0.5683**	**53.54** %	**72.65** %	**81.89** %
LDN-4	✓	0.2	✓	✓	0.5804	**6.9544**	**12.673**	0.6003	50.84 %	69.80 %	79.28 %
LDN-5	✓	0.2	✓		0.5865	7.0496	12.700	0.6026	50.71 %	69.64 %	79.09 %

[1] This column indicates whether the network training uses self-supervised learning (SSL). When *Teacher* networks are available, the relative pose prediction that constructs the photometric matching loss for the LDN is the prediction of *Teacher* networks.
[2] Model Distillation (KD). The LDN acts as a student network that imitates the outputs of the teacher depth network using $L1$ loss. This column also indicates whether *Teacher* network involved in training.

of Fig. 6. The BSM generates large yaw rates to guide the quadcopter away from the obstacles. However, in corner cases such as at t_2 and t_4, where the quadcopter is facing a planar wall or when the obstacle is in the center of view, the `manoeuvre` behavior cannot find a path. Therefore, the BSM switches to `redirect` mode to redirect the quadcopter to a safe direction. In situations such as at t_3, where obstacles are far from the quadcopter and the goal direction is directly in front, the drone moves forward and navigates towards the goal while steering at a low speed.



(a) Sparse environment (b) Dense environment

Fig. 4. Flight trajectories captured by Optitrack global motion capture system. Different color represents multiple trajectories with various starting points, with color goes deeper along the timestep. Obstacles are plotted manually for demonstration.

Corridor Experiments The quadcopter successfully navigated to the end of the corridor in most attempts during experiments in the corridor environment (demo video can be found in the supplementary material). However, in a few cases, the camera's field of view was blocked by a single obstacle after a sharp turn and the quadcopter crashed.

3.3 Network Generalization

In the right column of Fig. 6, we show examples of detecting objects that are not captured by the training set. The LDN being tested is trained on the *CyberZoo* dataset.

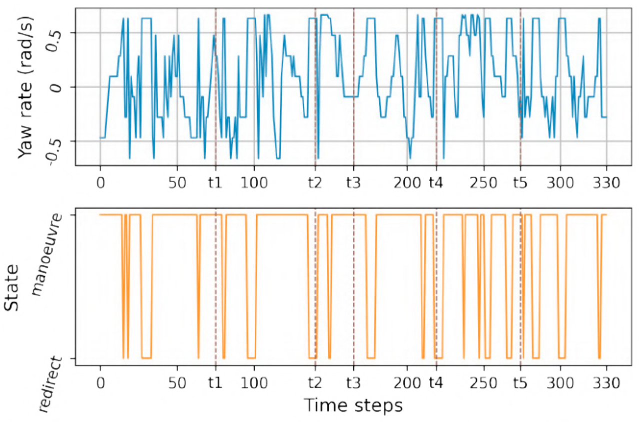


Fig. 5. Yaw rate and behavior state. By switching behavior state accordingly to the context, BSM generates suitable high-level control commands.

The top row depicts a successfully detected moving human with motion blur, even though humans remained stationary in the training set. The second row exhibits a partially detected obstacle. We hypothesize that the network's ability to generalize to new obstacles may be attributed to the familiar background of the environment. Despite that only a portion of an unknown obstacle is detected in many cases, our control strategy consistently generates appropriate actions. Additional unknown obstacles were placed in the *CyberZoo* environment, and the quadcopter successfully avoided them in multiple runs.

To test the generalization further, we utilized the same LDN trained in the *CyberZoo* to navigate the quadcopter in the *Corridor* environment. As displayed in the third row of the right column of Fig. 6, the network was able to detect a section of the wall, indicating a similar level of generalization as in the second row.

This may be due to the fact that the wall and floor structures also appear in the *CyberZoo* dataset. During

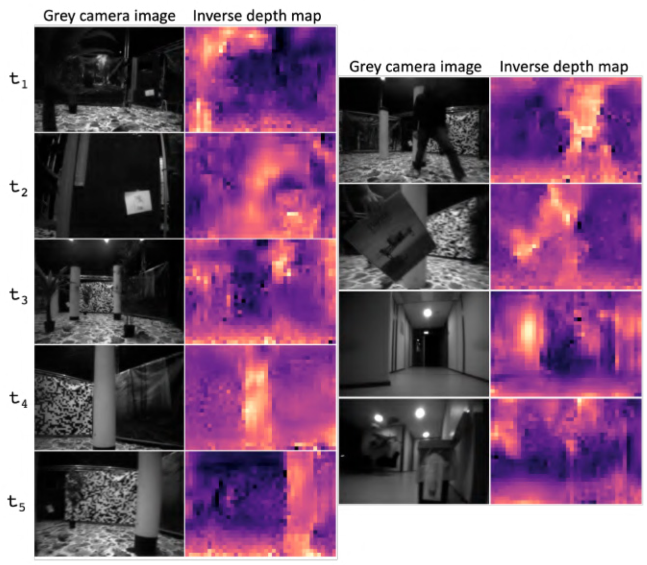


Fig. 6. Left column: Grey camera images and their corresponding inverse depth maps at the same timesteps as Fig. 5, i.e., $t_1 \sim t_5$. Behaviors during from $t_1 \sim t_5$ timesteps include turning right quickly, redirecting, turning left slowly, redirecting, and turning left quickly. Right column: Scenarios demonstrating the generalization capability of the system. Cases from top to bottom are dynamic obstacle (running human), unknown obstacle (cardboard box), unknown environment (corridor), and a failure case of detecting an unknown obstacle (trash can).

the experiment, the quadcopter flew safely through a part of the corridor until it crashed on an undetected trashcan, as shown in the fourth row of Fig. 6 (right column).

In this paper, we mainly focus on obstacle avoidance in environments that the LDN knows. The LDN is trained on a small dataset collected in the field. As for works Ranftl et al. (2020); Aleotti et al. (2020) that pursue outstanding generalization by exploiting huge training datasets, our network is not comparable. When the flight environment is unknown and thus network generalization has high priority, the workflow proposed in Aleotti et al. (2020) can be followed. We leave this to potential further works.

4. CONCLUSION

This paper presents a computationally efficient solution for nano quadcopter obstacle avoidance. Obstacle detection is conducted by a lightweight monocular depth network. The training schemes that do not require ground-truth labels are studied. We verify that the combination of reprojection-based self-supervised learning and knowledge distillation leads to the best accuracy. The control strategy is implemented by a behavior state machine that relies on the low-resolution depth map to detect and avoid obstacles, while balancing navigation efficiency and safety. In general, the whole system tackles the task well, featured by the low demand for computational power and the capacity of handling corner cases that are not solved by other solutions.

REFERENCES

Aleotti, F., Zaccaroni, G., Bartolomei, L., Poggi, M., Tosi, F., and Mattoccia, S. (2020). Real-time single image depth perception in the wild with handheld devices. *Sensors*, 21(1), 15.

Althaus, P. and Christensen, H. (2002). Behaviour coordination for navigation in office environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, 2298–2304 vol.3.

Bian, J., Li, Z., Wang, N., Zhan, H., Shen, C., Cheng, M.M., and Reid, I. (2019). Unsupervised scale-consistent depth and ego-motion learning from monocular video. *NeurIPS*, 32.

Bitcraze (2022). *Crazyflie 2.1 Specifications*. URL `https://www.bitcraze.io/products/crazyflie-2-1/`.

Bouwmeester, R.J., Paredes-Vallés, F., and de Croon, G.C. (2022). Nanoflownet: Real-time dense optical flow on a nano quadcopter. *arXiv preprint arXiv:2209.06918*.

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

Godard, C., Mac Aodha, O., Firman, M., and Brostow, G.J. (2019). Digging into self-supervised monocular depth prediction.

Liu, J., Li, Q., Cao, R., Tang, W., and Qiu, G. (2020). Mininet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 166, 255–267.

Loquercio, A., Maqueda, A.I., Del-Blanco, C.R., and Scaramuzza, D. (2018). Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2), 1088–1095.

Poggi, M., Tosi, F., Aleotti, F., and Mattoccia, S. (2022). Real-time self-supervised monocular depth estimation without gpu. *IEEE Transactions on Intelligent Transportation Systems*.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*.

Wagstaff, B., Peretroukhin, V., and Kelly, J. (2022). On the coupling of depth and egomotion networks for self-supervised structure from motion. *IEEE Robotics and Automation Letters*, 7(3), 6766–6773.

Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., and Scherer, S. (2020). Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4909–4916. IEEE.

Wang, Y., Li, X., Shi, M., Xian, K., and Cao, Z. (2021). Knowledge distillation for fast and accurate monocular depth estimation on mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2457–2465.

Yucel, M.K., Dimaridou, V., Drosou, A., and Saa-Garriga, A. (2021). Real-time monocular depth estimation with sparse supervision on mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2428–2437.