Monocular Depth Estimation for Drone Obstacle Avoidance in Indoor Environments

Haokun Zheng[1], Sidhant Rajadnya[1] and Avideh Zakhor[1]

***Abstract*— Autonomous nano-quadcopters possess large potential for indoor use. Existing works on autonomous flight however rely on large amounts of compute, therefore resulting in heavy and bulky platforms that can only be deployed outdoors. We present a monocular depth estimation method for autonomous obstacle avoidance and waypoint navigation of nano-quadcopters indoors demonstrated on the Bitcraze Crazyflie 2.1 which weighs a mere 33g. Our depth estimation model has 1.56 million parameters and is 4 MB, which after quantization becomes 1 MB. We transmit the images via WiFi from the onboard grayscale camera on the Bitcraze to a laptop, which then runs the 1 MB quantized model to generate small-size depth maps. Subsequently, we run our navigation algorithms on a laptop and transmit high-level motion commands back to the drone. We demonstrate obstacle avoidance capability of this end-to-end system through real-world flights in a variety of indoor environments.**

I. INTRODUCTION

Autonomous drones have gained significant traction in recent years, with a variety of applications ranging from aerial mapping and surveillance to sensing or search and rescue missions. Different environments and use cases necessitate vastly different hardware and software configurations. Specifically for indoor flights, the constraints of a denser environment limit the size and weight of the drone while simultaneously requiring higher precision and computationally efficient obstacle avoidance and navigation algorithms. In order to unlock the potential reduced risk of injury and increased responsiveness, the drone ideally needs to be able to autonomously navigate through an environment with minimal human intervention. So far most existing works on autonomous drone navigation and obstacle avoidance however mainly focus on larger drones with powerful hardware and sensor arrays, such as GPS, LIDAR, and multiple cameras. It is therefore imperative to develop efficient and lightweight systems that can enable autonomous flight on compact and low-compute hardware platforms without requiring diverse sensor inputs. In this work, we propose a low compute monocular vision-based autonomous obstacle avoidance and waypoint navigation flight system for a nano-quadcopter, which is designed to operate in indoor environments with limited computational resources and sparse sensory data. We use the Bitcraze Crazyflie 2.1 nano-quadcopter as experiment demonstration platform shown in Fig. 1 and pair the drone with two extension PCBs: 1) the AI Deck, which provides a monocular grayscale camera, an ultra-low power GAP8 RISC-V based microprocessor with 9 compute cores and an ESP32 WiFi AP and 2) the Flow Deck v2 extension board, supplying an downwards facing ToF sensor and an optical-flow-based positioning system. This allows us to design and deploy a system capable of autonomous flight on a platform, that weighs merely 33g and spans 11cm in diameter. At the center of our system lies a monocular depth estimation model, which is designed to be lightweight and computationally efficient, and is tailored to the specific 8-bit integer quantization constraints of the nano-quadcopter platform. We further demonstrate the efficacy of our depth estimation model by implementing an 1) obstacle avoidance flight algorithm and 2) a waypoint-oriented navigation flight algorithm in a series of real-world experiments. In the following, we first review related work in the field of autonomous drone flight and monocular depth estimation, and then detail the architecture and training of our monocular depth estimation model. We then describe the design and implementation of our obstacle and waypoint navigation algorithms and finally present the results of our real-world experiments and discuss the significance of our work.

Fig. 1: The Bitcraze Crazyflie 2.1 facing away from the camera with the AI Deck and Flow Deck v2 attached.

II. RELATED WORK

A. Autonomous Drone Flight & Obstacle Avoidance

A multitude of different approaches have been investigated to enable autonomous flight and obstacle avoidance for a drone. Currently, the two most common ways to implement autonomous flight and obstacle avoidance on a drone are either classical perception [1], [2], planning [3], [4] and control [5], [6] systems, or end-to-end deep RL-based

[1]Haokun Zheng, Sidhant Rajadnya and Avideh Zakhor are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, 253 Cory Hall, Berkeley, CA 94720, USA haokun.zheng@berkeley.edu, srajadnya@berkeley.edu, avz@berkeley.edu

approaches [7]–[9]. Latter approaches have demonstrated impressive ability to learn complex flight behaviors and obstacle avoidance strategies. This includes the ability to fly complex acrobatic trajectories [7] as well as the ability to fly at high speeds through cluttered outdoor environments such as forests [9] and high-speed drone racing [8]. These approaches however all rely on order of magnitudes heavier and larger quadcopter platforms and are intended for outdoor or large empty indoor environments with heavy onboard compute such as a Jetson TX2, which is not suited for tight indoor environments. One example is Loquercio et al. [10] who demonstrated the ability to leverage existing driving footage to train an end-to-end model that outputs a simple steering angle and collision probability for a drone. This closed-loop approach however is not only limited to outdoor environments, but also not capable or modifiable for waypoint-oriented flight. Yang et al. [11] implements a depth prediction and obstacle avoidance system on a drone and also deploys it in a real-world indoor environment. The used custom built drone which houses the required Intel NUC and Jetson AGX Xavier compute onboard however measures 60cm in width excluding rotors, making the system infeasible for safe real-world indoor use. Related works that build on the Bitcraze Crazyflie 2.1 or comparable size platforms either focus on less compute-intensive tasks such as localization [12], [13], other sensing tasks [14] unrelated to flight control, or obstacle avoidance [15], [16]. Palossi et al. [17] implemented a custom PULP expansion board to the Crazyflie 2.1 to enable a deep neural network-based obstacle avoidance flight for nano-drones, however the system's ability to avoid obstacles is only theoretically analyzed based on timing of a stop signal. Finally, none of the above mentioned works demonstrate the ability to implement waypoint-oriented flight on a nano-quadcopter, which is a crucial ability for many indoor drone applications such as indoor monitoring and security or indoor space mapping.

B. Monocular Depth Estimation

Monocular or single image depth estimation has emerged as a pivotal task in computer vision, facilitating various applications such as autonomous navigation, augmented reality, and scene understanding. Single Image Depth Estimation (SIDE), or more specifically, Relative Depth Estimation (RDE), a task predicting the relative distances between objects in an image, producing a depth map with farther objects appearing brighter and closer objects darker, are particularly challenging problems. As we aim to infer spatial depth information for every single pixel in an image, we require an output dimensionality of thousands of pixels mapped from 0-255 and a model that learns robust features to extract accurate depth information from a variety of potential image representations of the same depth space. Some of the first explorations into SIDE started with Saxena et al. [18], who utilized hand-crafted features and Markov Random Fields to generate relative depth maps for arbitrary image inputs. Modern approaches to SIDE involve deep-learning multi-layer CNNs and training routines using large-scale datasets such as NYUv2 [19] and SUN RGB-D [20].

Notably, seminal works like Eigen et al.'s [21] pioneering use of CNNs for depth prediction laid the groundwork for subsequent advancements. In this model, a two-tiered approach was utilized by running input images through coarse and fine networks to extract depth features. Following this, Laina et al. [22] constructed a new model based off of ResNet-50 [23]. While containing fewer parameters due to the model's fully convolutional architecture, overall context is retained through use of residual connections. The invention of the Transformer [24] and subsequently the Vision Transformer [25] gave rise to a new suite of SIDE models whose accuracy quickly made them the state of the art (SotA). Specifically, the Dense Prediction Transformer (DPT) [26] model utilizes Vision Transformers as its backbone, and adds a custom head to aid in SIDE. While the aforementioned models have excellent accuracy, their large size and complexity make their use in low-compute embedded systems challenging. Thus, models such as FastDepth [27] have gained prominence due to their compact architecture and reduced computational demands. Through its fully convolutional architecture and usage of lightweight encoder models like MobileNet [28], FastDepth [27] retains strong accuracy while making deployment on embedded systems possible.

III. METHODOLOGY

While many systems with relatively high complexity exist to implement autonomous flight of an UAV, our proposed system enables autonomous indoor obstacle avoidance and waypoint-based flight of a nano-quadcopter with a lightweight inference pipeline and sparse sensory data consisting of a monocular camera, a optical flow relative positioning system and an IMU. In order to accomplish such a task, we divide the problem into two main steps: We first generate a depth map from the onboard monocular camera frame through our SIDE model on a laptop and then subsequently supply the depth information to respective obstacle avoidance and waypoint-based flight navigation algorithms running on a laptop that yield high-level flight action commands which our Crazyflie platform can execute. Communication between the Bitcraze and the laptop is established via WiFi for the image stream, and radio for the high-level motion commands and radio connection. The obstacle avoidance and waypoint navigation algorithms are designed to be lightweight and computationally efficient given the specific constraints of the Crazyflie 2.1 nano-quadcopter platform and output either a required yaw rate or a desired forward velocity flight command. Fig. 2 provides an overview of the obstacle avoidance and waypoint-based flight system's respective steps and components.

A. Monocular Depth Estimation

1) Model Architecture: To achieve the desired task of efficient depth-estimation based autonomous obstacle avoidance, we propose a fully convolutional encoder-decoder architecture inspired by FastDepth [27], and is shown in the pink box in Fig. 2. We use a lightweight encoder featuring pointwise

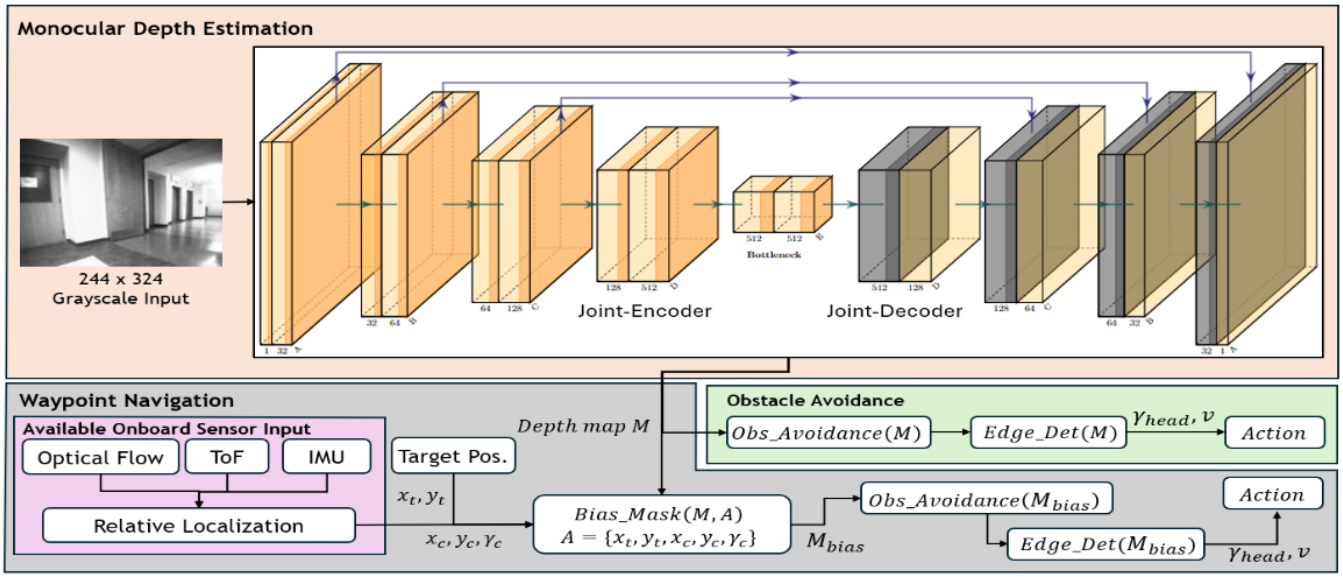


Fig. 2: System overview for depth-estimation based obstacle avoidance and waypoint navigation segmented into SIDE pipeline (orange), obstacle avoidance algorithm (green) and waypoint-based flight (gray) where $\{x_t, y_t\}$ are the goal coordinates, $\{x_c, y_c\}$ are the drone's position, and γ_c is the drone's heading. γ_{head} is yaw change and v forward velocity. $Obs\_Avoidance$ is illustrated in algorithm 1, $Edge\_Det$ refers to to algorithm 2, and $Bias\_Mask$ is defined in Equation 3.

and depthwise convolution blocks from MobileNet [28] to increase feature richness while minimizing excessive computational complexity. In Fig. 2, the "Joint-Encoder Block" is a multilayered block consisting of 5 sets of depthwise and pointwise convolutions, gradually increasing filter count from 128 to 512. Our decoder facilitates reconstruction of depth information from the encoded features through use of upsampling and convolution in a similar fashion to FastDepth [27]. The "Joint-Decoder Block" respectively consists of 5 sets of depthwise and pointwise convolutions, gradually decreasing filter count from 512 to 128. Before each convolution in the decoder, we additionally use a nearest-neighbor interpolation algorithm to upsample images according to sizes of the corresponding encoder layer and use residual connections between the encoder and decoder to aid in reconstruction.

Our approach diverges from many SotA architectures by deliberately imposing limitations on our model's size, constraining the number of parameters to 1.56 million. Consequently, the resolution of the resulting depth map is smaller by a factor of 4 in each dimension compared to grayscale input image. More specifically, the model was trained on image/label pairs of size (244×324) while the output depth-map is of size (62×82). This choice balances computational efficiency and depth estimation precision, facilitating integration into resource-constrained environments without compromising performance. Table I puts our model's size in perspective to SotA depth estimation models DPT-Large [26], DPT-Hybrid [26], Adabins [29] and the lightweight model FastDepth [27].

2) Dataset Acquisition: To train our depth estimation model, we assemble a diverse training dataset comprised of various indoor image datasets. In total, we train on 190,361 image-label pairs, with 50,000 coming from NYUv2 [19],

TABLE I: Model Binary Size Comparison

Model Name	Model Binary Size (MB)	Params. (Mil.)
DPT-Large	1290	343
DPT-Hybrid	470	123
Adabins	300	78.20
FastDepth	15	3.96
Ours	4	1.56
Our Quantized	1	1.56

70,496 from Stanford 2D-3D [30], 5065 from ICL-NUIMS [31], and 64,800 from Matterport3D [32]. The NYUv2 [19] dataset provides multiple indoor scenes captured from various perspectives, and also has well constructed and accurate ground truth values for the corresponding depth maps. Additionally, our depth estimation model employs a reponse-based knowledge distillation technique [33] to refine its training process, leveraging depth-maps generated by DPT-Large [26] as reference labels to guide the training of our model on all datasets except NYUv2 [19] to achieve consistency in training data resolution and dynamic range.

3) Training Specifications: We train our model with a 80/20 train-test split across the joint dataset over a total span of 500 epochs. Our loss function is constructed from a convex combination of Mean Squared Error (MSE) and Structural Similar Index Measure (SSIM) and defined as:

$$L(x,y) = \lambda * (1 - SSIM(x,y)) + (1-\lambda) * MSE(x,y) \quad (1)$$

where $\lambda \in [0,1]$. We set $\lambda = 0.8$. Choosing this value allows us to retain the significance SSIM gives to retaining structural representation while also minimizing pixel value discrepancy through use of MSE. Additionally, we retain numerical stability throughout the training process by normalizing both input and ground-truth depth images.

4) Deployment and Inference: We convert our model using TensorFlow Lite (TFLite) as it excels in providing

fast inference for deep-learning models while maintaining usable accuracy. To further compress the model's size and to match the GAP8 processor's environment, we apply a full 8-bit integer post-training static quantization, in which both model weights and activation layers are quantized to an 8-bit resolution, and parameter calibration is achieved through use of a representative dataset. During flight we establish a camera stream from the drone to a remote laptop where we run inference and subsequently return the algorithm's flight action commands through a radio link to the Crazyflie.

B. Obstacle Avoidance Flight Algorithm

A depth map-based obstacle avoidance flight algorithm especially tailored for our low-compute hardware constraints constitutes the core of our autonomous flight abilities. The obstacle avoidance flight algorithm fundamentally attempts to point the drone towards the deepest point in its view and is designed to be lightweight and computationally efficient for successful deployment on the GAP8 RISC-V processor post model inference. Algorithm 1 shows the high-level steps of the obstacle avoidance flight algorithm.

Algorithm 1: Obstacle Avoidance

Input:
Depth Map M
im_width W
forward_velocity v
camera_fov α
center_threshold_px t_{center}
min_head_chg β
edge_contrast_threshold $t_{edgeDet}$
edge_correction_constant c_{edge}
Algorithm:
while $no\_interrupt$ **do**
  $M_{cutout} \leftarrow cutout\_center(M, 10 \times 82)$;
  $indices_{candidates} \leftarrow max\_val\_indices(M_{cutout})$;
  $target \leftarrow closest\_to\_center(indices_{candidate})$;
  $target \leftarrow edge\_corr(target, t_{edgeDet}, M, c_{edge})$;
  $\gamma_{yaw} \leftarrow map\_index\_to\_angle(target, w, \alpha)$;
  $\gamma_{head} \leftarrow max(\beta, \gamma_{yaw})$
  if $target < \frac{W}{2} - t_{center}$ **then**
    $turn\_left(\gamma_{head})$;
  else
    if $target > \frac{W}{2} + t_{center}$ **then**
      $turn\_right(\gamma_{head})$;
    else
      $continue\_straight(v)$;
    end
  end
end

Configuring the drone to move at a predetermined constant height, the algorithm first extracts the center 10×82 pixels of the 62×82 depth map input as a reduced proxy for the drone's direct forward environment occupancy information. We subsequently calculate the ideal yaw angle off of the drone's current heading to steer the drone towards the deepest point in its view and therefore away from any close by obstacles. To accomplish this, we first search for the set of maximum values in the reduced frame cutout representing the deepest points $indices_{candidate}$, and hence all the desired headings, and then select the one positioned closest to the center to minimize turning movement. Once the desired depth index $target$ is identified, we then calculate the corresponding relative yaw change γ_{head} based on the $target$'s position in the width of frame and the drone's field of view α. To prevent the drone from making many small maneuvers without meaningful impact on the overall flight path, we furthermore apply a minimum yaw angle change β:

$$\gamma_{head} = \begin{cases} max(\frac{\alpha}{2}\frac{|target - \frac{W}{2}|}{\frac{W}{2}}, \beta), & |target - \frac{W}{2}| \geq t_{center} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where W is the width of the the depth map. To account for the depth-based approach's inherent tendency to cut turns close to corners and edges that sit right next to the deepest point in the depth map, we additionally design an edge correction algorithm to compensate and smooth out turns, explained shortly. Should the deepest point $target$ now be at least t_{center} pixels away from the center of the frame, then the drone turns according to γ_{head}, otherwise it will continue forward preventing unnecessary small turns. Table II shows the chosen parameters for the obstacle avoidance flight algorithm on our experimental platform.

TABLE II: Obstacle Avoidance Flight Algorithm Parameters

Parameter	Value
Depth map Image Width ($H \times W$)	62×82
Reduced Frame Cutout Size ($H \times W$)	10×82
Forward velocity v	0.4 m/s
Camera field of view α	96°
Center Threshold per side t_{center}	4 px
Min. Yaw Angle Change β	5°
Min. pixel contrast for edge $t_{edgeDet}$	35px
Edge Correction Pixel Shift c_{edge}	2px

1) Edge Detection and Yaw Angle Correction: One characteristic specific to indoor environments is that the deepest areas in the drone's view are often located right next to or partially occluded by an corner or edge, especially applying to hallway corners or general office space layouts. Leading the drone towards the deepest point in its field of view therefore often unnecessarily results in headings very close to corners and an increased collision risk. To allow for the depth map based obstacle avoidance algorithm to account for this, we include an edge detection algorithm that detects the presence of an edge next to the target point in the depth map and subsequently biases the intended yaw angle away from the edge, leading to more natural and smoother headings. Algorithm 2 illustrates this edge detection and correction processing step. The edge detection's sensitivity is determined by the deployed depth map model's specific contrast and controlled through the edge contrast threshold $t_{edgeDet}$. With $target$ representing the width-index with the brightest pixel in the depth map M at height j, we calculate the difference of brightness in between $M_{target,j}$ and the respective horizontal right and left values $M_{target\pm 2,j}$. Fig. 3 illustrates an depth map output containing an edge and the corresponding calculation checks for an edge left of $target$

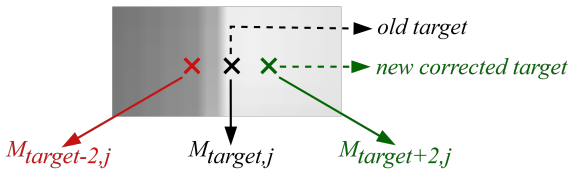


Fig. 3: Crop of example inference frame with identified edge left of target and subsequent $target$ correction by 2 pixels towards the right to the green position.

marked as black cross. In this example the difference in between $M_{target-2,j}$ and $M_{target,j}$ i.e. the depth difference between the red and black cross in Fig. 3, exceeds the edge contrast threshold defined by $t_{edgeDet}$ while the same does not hold for $M_{target+2,j} - M_{target,j}$ corresponding to the difference in between the black and green cross in Fig. 3. We thus identify an edge to the left and subsequently shift the target point for the yaw correction by 2 pixels away from the edge towards the right to avoid close headings to the edge. In the case in which we detect edges on both sides, the target point is left unchanged to not increase the risk of a collision.

Algorithm 2: Edge Detection and Correction

Input:
Depth Map M
im_width W
target_X_index $target$
edge_contrast_threshold $t_{edgeDet}$
edge_correction_constant c_{edge}
Algorithm:
$correct\_right \leftarrow false$;
$correct\_left \leftarrow false$;
if $|M[target]–M[max(0, target-2)]| > t_{edgeDet}$ **then**
  $correct\_right \leftarrow true$;
end
if $|M[target]–M[min(W–1, target+2)]| > t_{edgeDet}$ **then**
  $correct\_left \leftarrow true$;
end
if $correct\_right$ **then**
  if $correct\_left$ **then**
    $target \leftarrow target$;
  else
    $target \leftarrow max(target + c_{edge}, W–1)$;
  end
else
  if $correct\_left$ **then**
    $target \leftarrow min(0, target - c_{edge})$;
  end
end

C. Waypoint Navigation Flight Algorithm

Expanding on the obstacle avoidance flight algorithm, we develop an efficient waypoint-based navigation flight algorithm that not only enables the nano-quadcopter to avoid obstacles in an indoor environment but also simultaneously navigate T-junction turns towards a predetermined waypoint. This is realized by leveraging the Crazyflie's onboard optical flow-based relative position and heading estimates in conjunction with addition of a mask value n_j to each pixel in column j of the depth image to the inference depth map, biasing the drone's flight path towards the intended waypoint by artificially increasing the perceived spatial depth of the desired heading direction while preserving obstacle avoidance properties. An example bias mask is illustrated in Fig. 4 for a goal waypoint to the right of the drone outside of its field of view. The bias mask for flight towards a distinct waypoint given depthmap M with dimensions $H \times W$ is calculated as follows:

$$M_{i,j,biased} = M_{i,j} + n_j, \quad \forall 0 \leq i \leq H, 0 \leq j \leq W \quad (3)$$

where n_j is dependent on the angle γ between the drone's current heading and the vector pointing from the drone's current position to the goal waypoint. We define γ as negative should the goal waypoint be towards the left of the drone and positive for a goal waypoint towards the right. n_j is then calculated as follows:

$$n_j = \begin{cases} \min\left(max_{bias}, f_{bias} \times \frac{W-j}{W} \times \tan(\gamma)\right), & \text{if } \gamma < 0 \\ \min\left(max_{bias}, f_{bias} \times \frac{j}{W} \times \tan(\gamma)\right), & \text{if } \gamma \geq 0 \end{cases} \quad (4)$$

The max_{bias} parameter is a threshold value that limits the maximum bias that can be added to the depth map to prevent erasure of depth features, and the f_{bias} parameter is a scaling factor that determines the strength of the bias added to the depth map. Intuitively speaking, as seen in Fig. 4, the Mask bias function results in a straight ramp line at an angle γ until it reaches the maximum max_{bias}, after which it flattens to a constant value to prevent full erasure of the underlying spatial depth features. As the waypoint is to the right of the drone in the example in Fig 4, the ramp starts at 0 from the left side of the image and increases from left to right to incentivize the drone to head towards the waypoint.

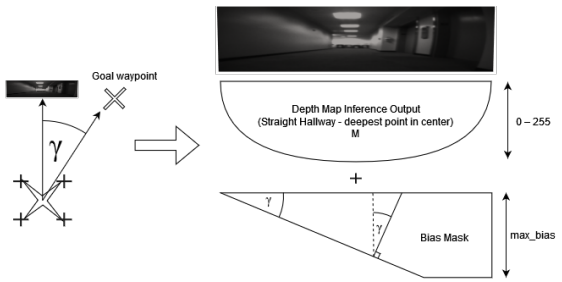


Fig. 4: Example bias mask added to inference depth map.

Should the drone approach a T-junction or a crossroad, the added bias mask component to the depth map will therefore influence the drone to prefer the path towards the waypoint by increasing that path's perceived spatial depth without erasing general depth map information and obstacle avoidance capabilities.

IV. EXPERIMENTS

We analyze our SIDE model performance as well as validate the autonomous obstacle avoidance and waypoint based flight navigation algorithms. While the SIDE model is qualitatively evaluated on its ability to accurately generate

depth maps, the obstacle avoidance and waypoint navigation flight algorithms are tested in previously unseen real-world controlled indoor environments. We use the Bitcraze Crazyflie 2.1 nano-quadcopter with the AI Deck and a Flow Deck v2 extension boards attached as our experiment platform. Leveraging the WiFi module on the Crazyflie 2.1, we are able to stream the onboard camera feed to a laptop and visualize the depth map and the flight path of the drone.

A. *Inference Analysis*

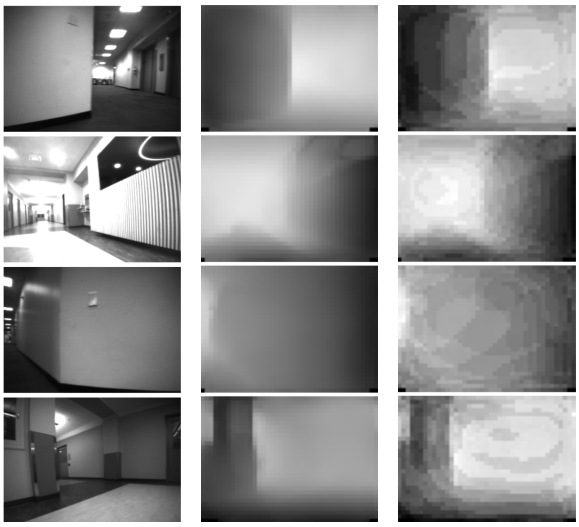
Our SIDE model proves to perform well on a variety of indoor scenes even given its tiny size. Across all inference outputs, the spatial depth correlates with pixel brightness. In Fig. 5, we provide sample inference frames extracted from our joint datasets for the fully int8-quantized as well as unquantized model respectively. The model consistently discerns the deepest point in each image, whether in quantized or unquantized states, as demonstrated by all example frames. Moreover, in the second and fourth images, the model not only identifies the deepest point but also detects a directional gradient associated with it. In Fig. 6, we analyze never before seen frames from the actual drone camera alongside their respective inference output obtained from our experimental settings. Once again, it is apparent that the model generalizes well to new and noisy environments and effectively identifies the deepest points in each frame, as well as any required directional depth-gradient.

(a) Input Frame (b) Ground truth (c) Unquant. Output (d) Quant. Output

Fig. 5: Various examples of SIDE on images from our aggregated datasets. Top-Down: NYUv2, Matterport3D, Stanford 2D-3D, ICL-NUIMS. Ground-truth for Matterport3D, Stanford 2D-3D, ICL-NUIMS generated via DPT-Large.

B. *Obstacle Avoidance Flight Algorithm*

We test the obstacle avoidance flight algorithm with the Crazyflie in a controlled indoor obstacle course environment. Fig. 7 shows the obstacle course setup and a top-down view is shown in Fig 9. The course is specifically challenging as the wooden boards and boxes are placed tightly together,



(a) Input Frame (b) Unquant. Output (c) Quant. Output

Fig. 6: Visualizations of our SIDE's model's performance on unseen test images. Frames were sourced from various local indoor hallway experiment environments.

Fig. 7: Obstacle course setup. Drone at starting position 2.

fully cover the background from the drone's point of view and have sharp corners that require a well tuned edge detection and correction algorithm to avoid collisions. In order to evaluate the obstacle avoidance flight algorithm, we let the Crazyflie repeatedly fly through the obstacle course from different starting positions for a total of 15 runs per configuration and observe its ability to autonomously avoid obstacles. Table III shows the success rate of the runs, demonstrating the system's ability to successfully navigate the obstacle course using both unquantized and quantized states of the model with the parameters shown in Table IV.

As expected, the unquantized model has a slightly higher success rate than the quantized one. Also, without edge correction the success rate is considerably lower than with correction. A video of a successful run of the drone in the obstacle course can be found here: `https://youtu.be/63pnnYmZl3g` The starting positions were specifically chosen with occluding objects in front of the start position to force the drone into obstacle avoiding maneuvers. Fig. 8 illustrate a sample raw camera frame as well as the corresponding inference depth map output from the int8-

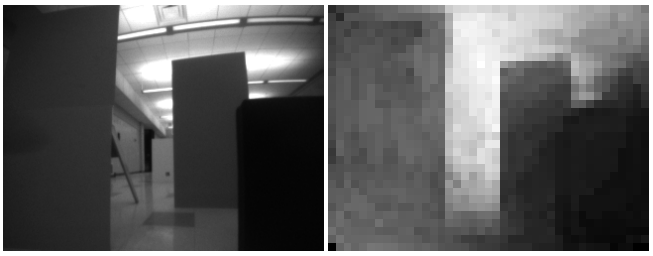
TABLE III: Obstacle Avoidance Experiment Success Rates

	Success Rate/Runs – Percentage			
Starting Position:	quantized + no edge corr.	unquantized + no edge corr.	quantized + edge corr.	unquantized + edge corr.
1	0/3 – 0%	0/3 – 0%	4/5 – 80%	5/5 – 100%
2	1/3 – 33%	0/3 – 0%	5/5 – 100%	5/5 – 100%
3	0/3 – 0%	1/3 – 33%	5/5 – 100%	5/5 – 100%

TABLE IV: Experiment Algorithm Parameter Configuration

Parameter	Value
Center Threshold per side t_{center}	4 px
Min. Yaw Angle Change β	5°
Min. pixel contrast for edge $t_{edgeDet}$	35px
Edge Correction Pixel Shift c_{edge}	2px
Bias Mask Maximum max_{bias}	128
Bias Mask Scaling Factor f_{bias}	128

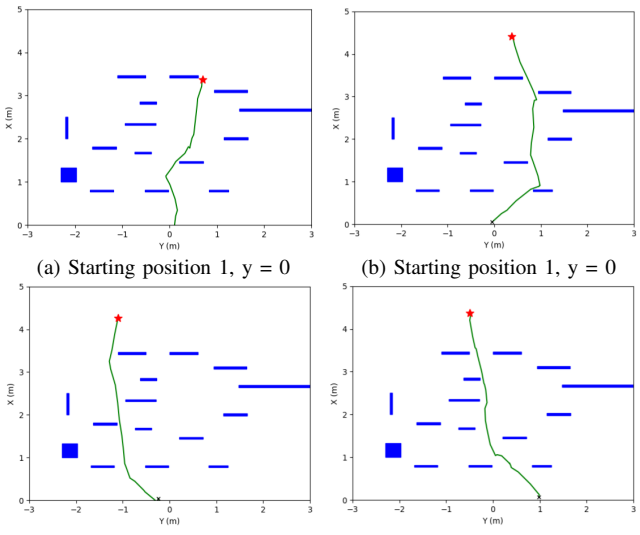
quantized model during an instance of the obstacle avoidance flight algorithm. It becomes clear, that the int8-quantized



(a) Camera grayscale input frame (b) Computed inference depth map

Fig. 8: Captured input frame and corresponding depth map.

instance of the model is able to generate precise-enough depth maps to clearly discern in between the obstacles at different depth levels. Fig. 9 shows the different starting positions and examples of flight paths observed during our experiment runs.



(a) Starting position 1, y = 0 (b) Starting position 1, y = 0

(c) Starting position 2, y = -0.2 (d) Starting position 3, y = 1

Fig. 9: Selection of flight paths at different starting positions for obstacle avoidance.

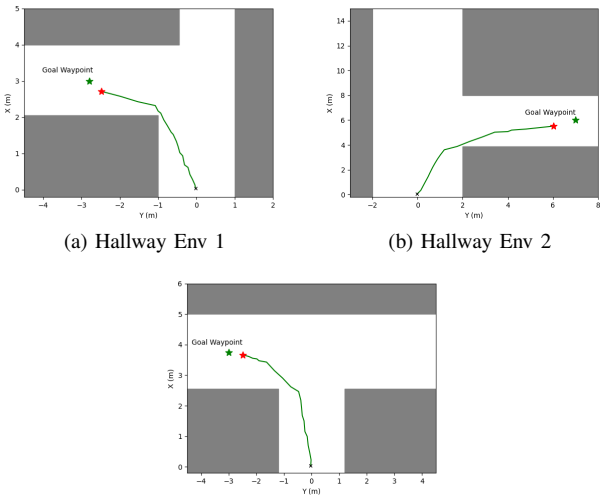
(a) Original frame (b) Inference result (c) Biased depth map

Fig. 10: Biasing of depth map through goal waypoint to the right outside of the drone's field of view.

C. Waypoint Navigation Flight Algorithm

We further analyze the waypoint flight algorithm with the drone in multiple previously unseen hallway T-junctions with differing geometries that would by themselves not cause the drone to head towards the waypoint. Fig. 10 shows a sample raw camera frame of the drone as well as the corresponding inference depth map output that is successfully biased towards the right due to the goal waypoint sitting at a 50° heading on the right side outside the visible frame.

We test the waypoint navigation flight algorithm with the Crazyflie drone and the quantized model running edge correction on three different hallway segment geometries and are able to navigate to our desired waypoints as intended. A video of a successful run of the drone in the hallway environment 1 can be found here: `https://www.youtube.com/watch?v=TOU0lIeJPug` Fig. 11 illustrates the environment layout as well as resulting flight paths. We observe



(a) Hallway Env 1 (b) Hallway Env 2

(c) Hallway Env 1

Fig. 11: Hallway geometries and flight path results for waypoint-based flight.

that the drone successfully heads towards the respective waypoints while avoiding the corner as obstacle solely based on the SIDE inference output and our fast algorithm.

V. CONCLUSION

We successfully demonstrate the ability for a nano-quadcopter solely equipped with a monocular camera and optical flow sensor to autonomously navigate through indoor environments using a lightweight depth estimation model and obstacle avoidance and waypoint navigation flight algorithms. Our compact depth estimation model is able to generate depth maps in real time and our obstacle avoidance and waypoint navigation flight algorithms are able to successfully guide the drone through a variety of indoor environments. Experiments show that the required integer quantization of the model for deployment on the Crazyflie 2.1 nano-quadcopter slightly impacts the algorithm's reliability. Priority when choosing or developing hardware platforms could therefore be given to those allowing for more precise quantization or even floating point operations.

REFERENCES

[1] Vlad-Cristian Miclea and Sergiu Nedevschi. Monocular depth estimation with improved long-range accuracy for uav environment perception. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2021.

[2] Michaël Fonder, Damien Ernst, and Marc Van Droogenbroeck. M4depth: Monocular depth estimation for autonomous vehicles in unseen environments. *arXiv preprint arXiv:2105.09847*, 2021.

[3] Junseok Lee, Xiangyu Wu, Seung Jae Lee, and Mark W Mueller. Autonomous flight through cluttered outdoor environments using a memoryless planner. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1131–1138. IEEE, 2021.

[4] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. Uav path planning using global and local map information with deep reinforcement learning. In *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021.

[5] Vemema Kangunde, Rodrigo S Jamisola Jr, and Emmanuel K Theophilus. A review on drones controlled in real-time. *International journal of dynamics and control*, 9(4):1832–1846, 2021.

[6] Mohamed Okasha, Jordan Kralev, and Maidul Islam. Design and experimental comparison of pid, lqr and mpc stabilizing controllers for parrot mambo mini-drone. *Aerospace*, 9(6):298, 2022.

[7] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. *arXiv preprint arXiv:2006.05768*, 2020.

[8] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212. IEEE, 2021.

[9] Yunlong Song, Kexin Shi, Robert Penicka, and Davide Scaramuzza. Learning perception-aware agile flight in cluttered environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1989–1995. IEEE, 2023.

[10] Antonio Loquercio, Ana I Maqueda, Carlos R Del-Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[11] Xin Yang, Jingyu Chen, Yuanjie Dang, Hongcheng Luo, Yuesheng Tang, Chunyuan Liao, Peng Chen, and Kwang-Ting Cheng. Fast depth prediction and obstacle avoidance on a monocular drone using probabilistic convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 22(1):156–167, 2019.

[12] Stefano Bonato, Stefano Carlo Lambertenghi, Elia Cereda, Alessandro Giusti, and Daniele Palossi. Ultra-low power deep learning-based monocular relative localization onboard nano-quadrotors. *arXiv preprint arXiv:2303.01940*, 2023.

[13] Shushuai Li, Christophe De Wagter, and Guido CHE De Croon. Self-supervised monocular multi-robot relative localization with efficient deep neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9689–9695. IEEE, 2022.

[14] Bardienus P Duisterhof, Shushuai Li, Javier Burgués, Vijay Janapa Reddi, and Guido CHE de Croon. Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9099–9106. IEEE, 2021.

[15] Cheng Liu, Erik-Jan van Kampen, and Guido CHE De Croon. Adaptive risk-tendency: Nano drone navigation in cluttered environments with distributional reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[16] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 international conference on robotics and automation (ICRA)*, pages 6008–6014. IEEE, 2019.

[17] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. A 64-mw dnn-based vifsual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 6(5):8357–8371, 2019.

[18] Ashutosh Saxena, Sung Chung, and Andrew Ng. Learning depth from single monocular images. *Advances in neural information processing systems*, 18, 2005.

[19] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012.

[20] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[21] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[22] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2020.

[26] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[27] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019.

[28] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[29] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021.

[30] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[31] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[32] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[33] Chuanguang Yang, Xinqiang Yu, Zhulin An, and Yongjun Xu. Categories of response-based, feature-based, and relation-based knowledge distillation. In *Advancements in Knowledge Distillation: Towards New Horizons of Intelligent Systems*, pages 1–32. Springer, 2023.