Obstacle Avoidance of a UAV Using Fast Monocular Depth Estimation for a Wide Stereo Camera

Euihyeon Cho, Hyeongjin Kim, Pyojin Kim, and Hyeonbeom Lee

***Abstract*—In this study, we designed an obstacle avoidance algorithm for a quadrotor unmanned aerial vehicle (UAV) equipped with a wide field-of-view (FOV) stereo camera, utilizing a learning-based depth estimation approach. Depth estimation using monocular cameras is gaining interest as a viable alternative to large and heavy sensors, such as light detection and ranging (LiDAR) sensors. However, deep learning-based depth estimation has low accuracy unless the depth estimation is done in an environment similar to that of the training data. Therefore, we first designed a depth estimation network for a wide-FOV stereo camera using two cameras. Then, we estimated the depth image using a convolutional neural network and improved the accuracy using stereomatching. We used the estimated depth images to develop a simple behavior-arbitration-based control algorithm that steers the quadrotor away from 3-D obstacles. We conducted simulations and experiments using a real drone in an indoor and outdoor environment to validate our proposed algorithm. An analysis of the experimental results showed that the proposed method could be employed for navigation in cluttered environments.**

***Index Terms*—Depth estimation, obstacle avoidance, quadrotor, wide stereo camera.**

Manuscript received 25 December 2023; revised 19 March 2024 and 5 June 2024; accepted 9 July 2024. This work was supported in part by the Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and Unmanned Vehicle Advanced Research Center funded by the Ministry of Science and ICT, the Republic of Korea under Grant NRF-2020M3C1C1A01086411; and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant RS-2023-00213897. *(Corresponding author: Hyeonbeom Lee.)*

Euihyeon Cho and Hyeongjin Kim are with the School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea.

Pyojin Kim is with the School of Mechanical and Robotics Engineering, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, South Korea.

Hyeonbeom Lee is with the Department of Mechanical Engineering, Ajou University, Suwon 16499, Republic of Korea (e-mail: hbeomlee@ajou.ac.kr).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TIE.2024.3429611.

Digital Object Identifier 10.1109/TIE.2024.3429611

I. Introduction

UNMANNED aerial vehicles (UAVs), which are highly maneuverable in 3-D space, have been recently employed in various fields, such as subterranean exploration [1], delivery [2], [3], and mapping [4]. The safe navigation of UAVs has been studied using visual cameras [5], [6], [7], [8], 3-D light detection and ranging (LiDAR) sensors [1], and event cameras [9]. However, these sensors have serious drawbacks such as high cost [1], [9] or collisions caused by a limited field-of-view (FOV) [5], [6], [7].

In this article to handle the aforementioned problems, we leverage recently developed monocular camera-based depth estimation. This algorithm has attracted significant attention for its ability to accurately obtain a dense depth image in both indoor and outdoor settings using an inexpensive monocular camera [10], [11]. Monocular depth is adaptable for use with fisheye cameras [12] or in extreme weather conditions through the utilization of thermal imaging cameras [13]. Especially when using a fisheye camera, accuracy may be degraded due to differences in image size from the dataset and image distortion issues. In addition, while the existing feature-based depth estimation techniques [14] exhibit poor performance in the presence of dynamic obstacles, monocular depth estimation (MDE) demonstrates satisfactory performance even in dynamic environments. Therefore, one of our objectives in this article is to improve the accuracy of MDE, even in unseen environments.

A. Contribution

The main contributions of this study are as follows.

1) We developed a fast depth estimation network to enhance the computational speed and accuracy of the MDE. To enhance depth estimation accuracy, we designed a depth-refinement algorithm, which significantly improves estimation performance. We performed a performance comparison with existing algorithms [10], [11], [15], [16], [17] using various datasets including Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI), Virtual KITTI2, ApolloScape, dense depth for autonomous driving (DDAD), and our outdoor dataset. The proposed algorithm shows fast and consistent performance even in unseen and outdoor environments.
2) We perform comprehensive depth-estimation performance evaluation on more diverse datasets such as

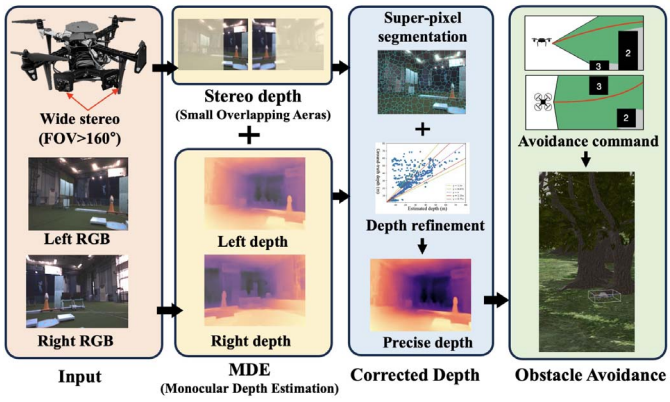


Fig. 1. Integrated framework of our autonomous system.

KITTI, Virtual KITTI2, ApolloScape, DDAD, and our collected dataset. This analysis demonstrated that our algorithm accurately estimates depth across various environments.

3) We conducted an obstacle avoidance experiment utilizing depth estimation technology using our proposed algorithm as shown in Fig. 1. Our algorithm generates avoidance commands solely based on estimated depth images and does not require additional mapping [4], [8] or obstacle avoidance datasets [6]. In addition, unlike the approach proposed in [5], our method can effectively handle various complex types of obstacles.

II. Related Works

Computational load and complexity reduction in obstacle avoidance when using optical flows have been extensively studied [18], [19], [20]. Flow-based algorithms are developed on the premise that insects successfully avoid obstacles though their computation complexity is much lower than that of trajectory optimization [3], [8]. However, flow-based obstacle avoidance algorithms [18] cannot be applied to complex environments without accurate depth estimation. On the other hand, stereo-based algorithms [19], [20] can deal with the depth problem. In [19], stereo-based depth estimation was employed to compute the optical flow. In [20], for long-range depth measurement, a variable baseline stereo camera was used. Nevertheless, flow-based algorithms [18], [19] are not suited for long-range obstacle avoidance, and the method in [20] suffers from the limited FOV of the stereo camera.

Earlier studies have proposed autonomous flight for a quadrotor with a fisheye monocular camera [8] or a wide stereo camera [4] using a monocular visual-inertial system, but it is difficult to deal with dynamic obstacles when using feature-based obstacle avoidance. Learning-based autonomous navigation algorithms have been developed to overcome dynamic obstacles and other long-range avoidance problems [6], [7]. In [6], [7], an autonomous drone could generate the steering command to achieve a safe flight in outdoor environments using a residual convolution architecture. However, in these studies, obstacle avoidance performance is not ensured in an unlearned environment

To overcome this problem, an obstacle avoidance algorithm that used learning-based depth estimation was proposed in [5]. It employed a MDE algorithm based on the approach introduced in [21]. The desired heading angle and the forward/backward speed of the UAVs were calculated from an estimated depth image. However, these methods [5] also result in a limited FOV for obstacle detection, and avoidance performance may be degraded because of the low accuracy of depth estimation. To improve the quality of depth estimation and overcome the limited FOV, learning-based depth estimation using a fisheye camera [22] was studied; however, depth could not be estimated in real-time in this study. Learning-based depth estimation using a wide-FOV monocular camera [23] was used for vehicle parking; however, owing to its low accuracy, it is difficult to adapt it to various environments.

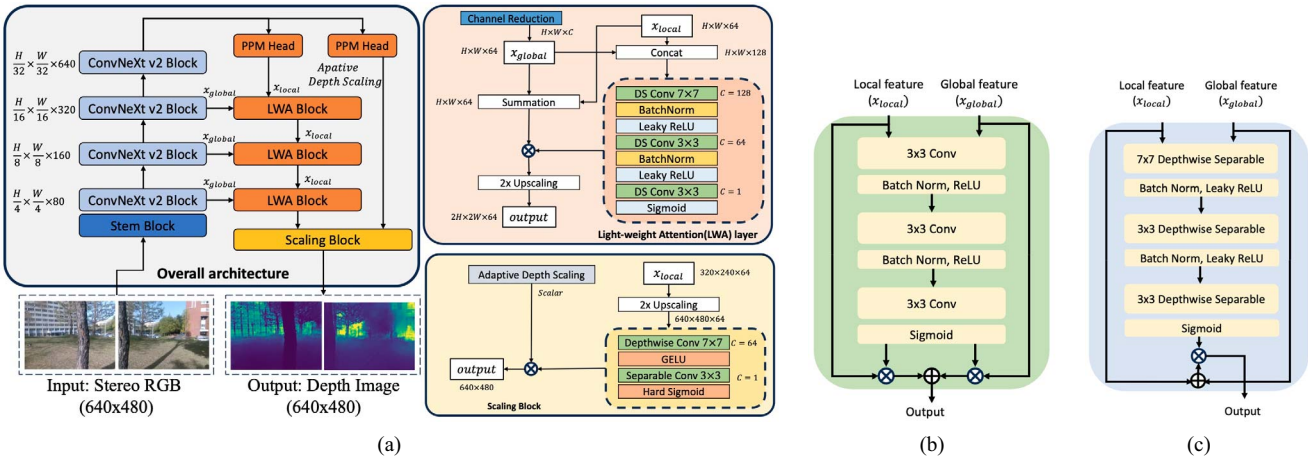
III. Depth Estimation

The objective of MDE is to obtain a high-quality and dense depth from a monocular image using a deep learning network. Eigen et al. [21] first proposed MDE using deep learning. Since then, many studies have been carried out to improve the overall depth estimation [10], [11], [15], [16], [17], [24]. Recently, following the success of the Transformer in language processing, the visual transformer has been applied for depth estimation [10], [11], [15], [16]. Despite their high accuracy on the trained dataset, these methods [10], [11], [15], [16] are computationally intensive and exhibit inaccurate estimation performance on unseen datasets. To address these limitations, a recent approach [17] has been developed based on zero-shot transfer learning using multiple depth estimation accuracy with single network (MiDAS) [25]. However, this method also remains computationally heavy, and their estimation performance in completely different environments is unreliable. Therefore, in this study, we aim to overcome these challenges by developing a fast and reliable depth estimation method even for unseen environments.

A. Monocular Depth Estimation

The main objective of our network is to predict a pair of depth images $\hat{\mathbf{d}} \in \mathbb{R}^{H\times W\times 1}$ from a given pair of RGB images $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$. To achieve this goal, we first design our learning model as illustrated in Fig. 2(a). Herein, our depth estimation network consists of encoder, decoder, and skip connections for the global feature. Inspired by [26], we utilized the encoder block employing the ConvNeXt v2 model. The encoder learns the global dependencies of input images. The stem block in the encoder downsamples the input images to a proper feature map size. The stem block comprises a 4×4 convolutional layer with a stride of 4, which results in a $4\times$ downsampling of the input image. The key advantage of this encoder is its ability to decrease the time needed for image patching, a process commonly used in existing visual transformers.

To reduce the computational payload on the decoder, we proposed the decoder with a lightweight attention (LWA) layer and scaling block [Fig. 2(a)]. In the LWA layer, we employed depthwise convolution, which was originally introduced in [27]. Depthwise convolution can be processed faster than



(a) (b) (c)

Fig. 2. Learning model for the depth estimation. (a) Our proposed model with light-weight attention (LWA) block and scaling block. (b) Conventional layer [15]. (c) Simplified our LWA layer.

conventional convolutions, but performance may decrease accordingly. In conventional algorithms such as [15], a 3×3 convolutional layer with multiple multiplications was utilized in the decoder [(Fig. 2(b)]. In our article, we also adopted a relatively large depthwise convolution layer with dimensions of 7×7 in the first decoder to better capture local dependencies, employing a single multiplication. Additionally, we integrated the pyramid pooling module (PPM) head structure from [28], which assists in learning appropriate local dependencies for depthwise convolution.

We analyze the computation time for each layer. The f_t^{conv}, total computation of conventional layer in Fig. 2(b) with C input channels, K Kerner size and image size $H \times W$ can be written as follows:

$$f_t^{\text{conv}} = \text{CHW} \times K^2 \times (2.5C + 1) + 3\text{CHW}$$

where $K = 3$. f_t involves the computation for 3×3 convolution, batch norm, rectified linear unit (ReLU), both multiplication and addition at the end of the convolution process in Fig. 2(b). The f_t^{our}, total computation of our LWA layer in Fig. 2(c) can be written as follows:

$$f_t^{\text{our}} = \text{CHW} \times (2K_1^2 + 1.5K^2 + 2.5C + 0.5) + 2\text{CHW}$$

where K_1 is 7 on the first depthwise convolution and $K = 3$. For example with $K_1 = 7$, $C = 64$, $f_t^{\text{conv}} = C \times H \times W \times 1452$. However, our algorithm only requires $f_t^{\text{our}} = C \times H \times W \times 274$, making our proposed algorithm approximately five times faster than the conventional algorithm.

The major limitation of the state-of-the-art algorithms including NewCRFs [11] and global-local path network (GLP) algorithm [15] lies in its fixed estimation range, which is constrained to a maximum distance of 80 m in the KITTI dataset. In both algorithms [11], [15], the depth is computed by scaling the output of the deep learning network by a factor of 80. To address this issue, we have developed a novel scaling block to overcome this limitation [Fig. 2(c)]. Due to this structure, our depth model can learn from diverse datasets with varying maximum depths.

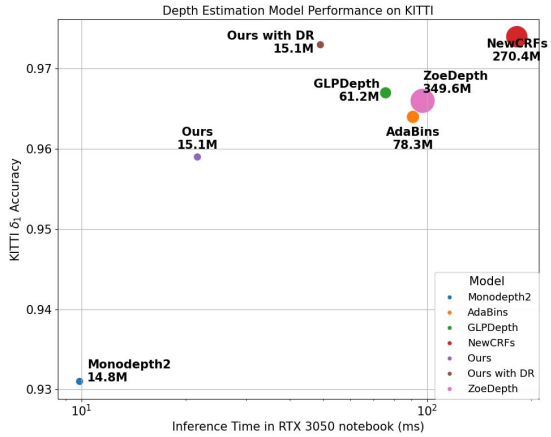


Fig. 3. Scatter plots depicting algorithm efficiency, measured by estimation performance on the KITTI dataset, versus inference time. The size of each circle corresponds to the number of model-network parameters.

For the loss function, inspired by [21], we used the scale-invariant (SI) loss function for our deep-learning network as

$$L = \alpha \sqrt{\frac{1}{T}\sum_{i=1} g_i^2 - \frac{\lambda}{T^2}\left(\sum_{i=1} g_i\right)^2} \tag{1}$$

where $g_i = \log(\hat{d}_i) - \log(d_i)$, and T is the number of pixels. $\hat{d}_i$ be a predicted depth and let d_i be the ground-truth depth. $\alpha = 10$, $\lambda = 0.85$ are the user-defined parameter.

Parameter comparison on deep-learning models can be shown in Fig. 3 which illustrates both the estimation accuracy on the KITTI dataset and the time required to process a single image. Computation time was measured on a laptop equipped with an Intel i7-13700H processor, 16GB memory, and an RTX 3050 mobile. The detailed computational time of our method is described in Table I. In contrast to the state-of-the-art algorithms such as NewCRFs [11] (about 181 ms per image) or Zoedepth [17] (about 96 ms per image), which demonstrate excellent performance, our algorithm with depth refinement achieves outstanding performance with fast computation time.

TABLE I
AVERAGE COMPUTATION TIME OF OUR PROPOSED METHOD

	Monocular Depth	Stereo Matching	Super Pixel	Scaling	Two Image Depth (Total)	Avoidance Command
Laptop	42.9[ms]	32.4[ms]	21.7[ms]	1.2[ms]	98.2[ms]	15.0[ms]

Due to the simplified network structure and reduced computational complexity, we were able to design a network model that is up to nine times lighter than existing algorithms. Further detailed analysis will be presented in Section V.

B. Depth Refinement

Learning-based depth estimation has low accuracy when the real-world scene is different from the training data, whereas stereomatching-based depth estimation does not easily yield a dense depth image and is unreliable owing to image texture.

We developed a depth refinement algorithm that matches learning-based depths with stereomatching-based depths for accurate depth estimation. Owing to our camera structure, we first perform a rectification process on the input images for stereomatching [Fig. 4(a)]. Stereomatching-based depth estimation has poor accuracy for an image with poor texture. Therefore the estimated value in a poor-texture area is excluded during depth refinement, and the superpixel in the RGB image, which is an algorithm for grouping perceptually similar pixels, is used to select reliable depths. In our article, we utilized the Libelas [29] for stereomatching.

Inspired by [25], the output of our depth estimation can be corrected as

$$\mathbf{d} = s(\mathbf{d})\hat{\mathbf{d}} + t(\mathbf{d})$$

$$t(\mathbf{d}) = \text{median}(\mathbf{d}),\ > s(\mathbf{d}) = \frac{1}{n}\sum_{i=1}^{n}|d_i - t(\mathbf{d})|. \qquad (2)$$

However, (2) is difficult to implement in practice because $s(\mathbf{d})$ and $t(\mathbf{d})$ are computed using a ground-truth depth data only. To address this problem, we propose a depth refinement algorithm with linear regression, as described in Fig. 4(b). We estimate depths corresponding to the stereomatched depths in the overlapping area of the two images. Assuming that all pixels in the same superpixel group belong to the same area, the depth variation within the same superpixel will be small. Therefore, when selecting stereo depths, only high-reliability data, in which the change in the depth value within the same superpixel group is small, are selected. Moreover, any estimated depth values derived from the representative monocular camera within the superpixel group that deviated by more than 50% from the stereo depth were deemed outliers and consequently excluded from the regression analysis. The utilization of this superpixel algorithm is crucial for establishing a robust relationship between the estimated depth and stereo matching. As shown in Fig. 4(c), without filtering using superpixels, obtaining meaningful correlations becomes challenging. Conversely, employing superpixels facilitates the identification of meaningful correlations by effectively eliminating outliers.

Using the linear regression algorithm, we estimate the coefficient in (2). For simplicity, let s_d and t_d be the estimated values

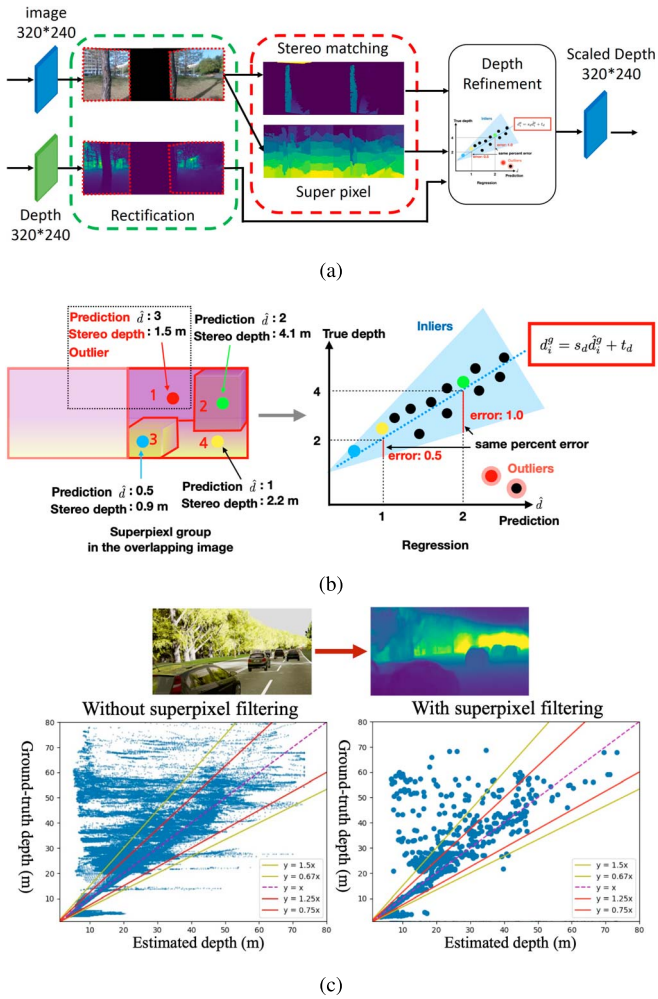


Fig. 4. Regression with learning-based depth prediction and stereo depth. (a) Depth refinement process. (b) Regression process. (c) Noise reduction using superpixel.

of $s(\mathbf{d})$ and $t(\mathbf{d})$, respectively. For the depth refinement, we denote the monocular depth of the ith pixel in the gth superpixel group (i.e., $\hat{d}_i^g$) and the refined depth corresponding to d_i^g (i.e., d_i^g). Using these terms, the refined depth can be calculated as follows:

$$d_i^g = s_d^f \times \hat{d}_i^g + t_d^f. \qquad (3)$$

Note that the update process to obtain s_d^f and t_d^f is iterative and performed for every input image. However, in certain situations, a valid stereomatching value required for the update might not be available. In such cases, the update should not be performed. Therefore, the update equation for s_d and t_d using inlier data can be written as follows:

$$\begin{cases} s_d = \dfrac{\sum d_i^g \sum(\hat{d}_i^g)^2 - \sum \hat{d}_i^g \sum d_i^g \hat{d}_i^g}{n\sum(\hat{d}_i^g)^2 - (\sum \hat{d}_i^g)^2} \\ t_d = \dfrac{n\sum \hat{d}_i^g d_i^g - \sum \hat{d}_i^g \sum d_i^g}{n\sum(\hat{d}_i^g)^2 - (\sum \hat{d}_i^g)^2}, & \text{if } n \geq \text{threshold} \\ s_d, t_d : \text{Do not update(Same as before)} & \text{otherwise} \end{cases}$$

where n is the number of valid data in the stereomatched depths. The s_d and t_d pass through a first-order filter with time-constant α for noise removal.

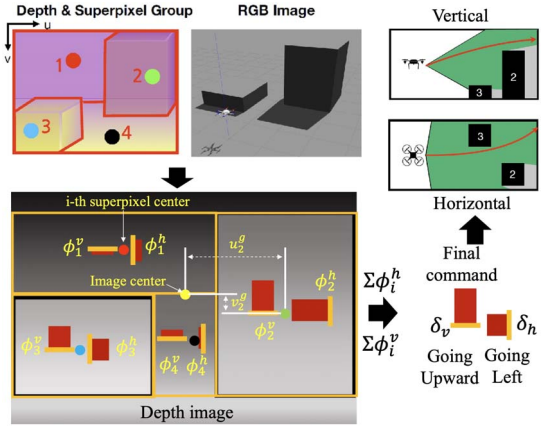


Fig. 5. Example of the desired velocity and heading angle command.

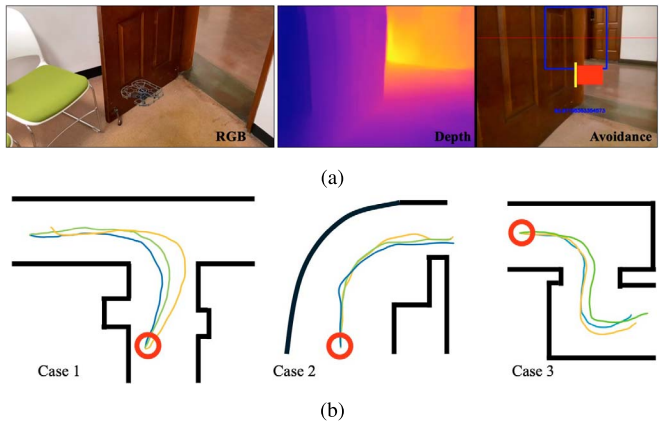


Fig. 6. Avoidance test on indoor environments: (a) Picture taken during flight (Case 2). (b) Test results.

IV. Autonomous Flight Algorithm

Our obstacle avoidance strategy is as follows: First, the image is divided into several superpixels. From each superpixel, we calculate the steering and thrust commands. Finally, the avoidance command for the drone is generated as illustrated in Fig. 5. It's crucial to note that when generating commands for each superpixel, we take into account the size of the superpixel in the depth image and its distance from the image center. These factors are essential in determining accurate and effective avoidance commands for the navigation of drones.

To achieve our objective, the steering and thrust commands can be computed

$$\phi_f^h = \sum_i^{S_n} \phi_i^h = \sum_i^{S_n} D_i^g W_i^h$$
$$\phi_f^v = \sum_i^{S_n} \phi_i^v = \sum_i^{S_n} D_i^g W_i^v \tag{4}$$

where $W_i^h = \exp(-(u_i^g/\text{HFOV})^2)$ is the horizontal weight for steering and $W_i^v = \exp(-(v_i^g/\text{VFOV})^2)$ is the vertical weight. S_n is the total number of superpixels in an image and the upper letter, g, denotes representative data in the superpixel group. u_i^g and v_i^g are the horizontal and vertical bearing angles in the image with respect to the image center. For example, if $u = 0$ and $v = 0$ in the image coordinates with a 110° (≈1.92 rad) horizontal FOV (i.e., HFOV) and a 74° (≈1.29 rad) vertical FOV (i.e., VFOV), $u_i^g = -0.96$ rad and $v_i^g = -0.645$ rad. Thus, W_i^h and W_i^v vary from 0.779 to 1.

In (4), we compute the exponential depth, denoted as D_i^g, rather than the direct depth represented as $-d_i^g$, in the ith superpixel group. The calculation for D_i^g can be expressed as follows:

$$D_i^g = e^{(d_c - d_i^g)} \frac{N^g S_n}{H_{\text{image}} W_{\text{image}}} \tag{5}$$

where H_{image} and W_{image} are the horizontal and vertical resolution of the image, respectively. d_i^g is the representative depth in the ith superpixel group and d_c is a user-defined parameter that causes the exponential value to be larger if d_i^g is smaller than d_c. N^g is the number of pixels in the gth superpixel group. For example, if an image with $H_{\text{image}} = 640 \times W_{\text{image}} = 480$ pixels has $S_n = 10$ superpixel group and each superpixel group has $N^g = 3072$ pixels, each superpixel group has the same weight: $(3072 \times 10)/(640 \times 480) = 0.1$. The use of exponential depth aims to assign a high weight to nearby areas while reducing the weight significantly for longer distances. Simultaneously, the weighting of superpixels (i.e., $N^g S_n / H_{\text{image}} W_{\text{image}}$) based on the number of pixels is done to generate commands in the opposite direction of those with larger volumes or areas.

By exploiting the exponential depth in (5) and (4), we can define steering angle δ_h and altitude rate δ_t respectively as

$$\delta_h = \phi_f^h \exp(-(\phi_f^h/G_a)^2), \;\; \delta_v = \phi_f^v \exp(-(\phi_f^v/G_a)^2) \tag{6}$$

where G^a is the user-defined gain. The exponential term in (6) is designed to smooth the command.

To analyze the performance of the obstacle avoidance algorithm δ_h, we conducted a simple experiment. In this experiment, we employed the depth estimation model trained on the NYU indoor dataset, and the depth refinement algorithm was not applied. The experimental results are presented in Fig. 6. Case 1 involved an environment where depth estimation was challenging due to walls lacking texture. Case 2 depicted passing through a narrow passage, and case 3 represented a curved driving scenario. During each 10 flight experiments in the same environment, successful avoidance was achieved in both cases 1 (10/10) and 2 (10/10). An 80% (8/10) avoidance performance was confirmed in case 3. This is due to the very short distance between obstacles in case 3, making it difficult to secure a proper viewing angle.

To arrive at the target point, we should compute the desired velocity v^d using the collision probability p_{coll}. Our flight algorithm is summarized in Algorithm 1. *get-steering-and-collision-prob* function computes steering angle rate δ_h, altitude rate δ_v, and collision probability p_{coll}. Here, G_c is the user-defined gain. In δ_{coll}, the term $(\phi_f^h + \phi_v^h)$ assigns a high weight in the center of the depth image because D_i^g varies from zero to one in (5), the denominator S_n normalized the probability. Therefore, if all depth values in an image are sufficiently small, the collision probability is close to one.

Algorithm 1: Our Method for Drone Control

Input: List of data $S = (d_i, u_i, v_i, S_n)$
Output: Command angle ψ_t, Velocity v_d
Function *get-steering-and-collision-prob*
  Input : $S = (d_i^g, u_i, v_i, S_n)$
  $(\phi_f^h, \phi_f^v) \leftarrow eq.(6)$, $(\delta_h, \delta_v) \leftarrow eq.(8)$
  $p_{coll} \leftarrow \delta_{coll} \cdot e^{(\delta_{coll} \cdot G_c)^2}$, $\delta_{coll} \leftarrow (\phi_f^h + \phi_f^v)/S_n$
  return δ_h, δ_v, p_{coll}
end
Function *command-for-desired-location*
  Input : $\delta_{h_l}, \delta_{h_r}, p_{coll_l}, p_{coll_r}$, Goal($P_g$), Current position($P_c$), Current time($t_{now}$)
  Data: Previous command angle ψ_{t-1}, Previous collision probability $\delta_{coll_{t-1}}$, Critical time t_c
  $p_{coll_t} \leftarrow$ **mean**$(p_{coll_l}, p_{coll_r}) \cdot 0.3 + p_{coll_{t-1}} \cdot 0.7$
  $v_d \leftarrow V_{max} \cdot (1 - p_{coll,t})$
  if *distance*(P_c, P_g) $>$ *Threshold* **then**
    $\delta_g \leftarrow$ *desired-heading*(P_c, P_g)
    $\delta_t \leftarrow \delta_{h_r} - \delta_{h_l}$
    $w, t_c \leftarrow$ *get-weight*(δ_t)
    $w_{time} \leftarrow e^{-0.05 \cdot (t_{now} - t_c)} \cdot w$
    $\delta_t \leftarrow w_{time} \cdot \delta_t + (1 - w_{time}) \cdot \delta_g$
    $\psi_t \leftarrow \psi_{t-1} \cdot 0.7 + \delta_t \cdot 0.3$
  else
    $v_d = 0$, $\psi_t = \psi_{t-1}$
  end if
  return ψ_t, v_d
end
Function *get-weight*(δ_t)
  if $|\delta_t| < 10°$ **then**
    return $w = 0.5$, t_c: *no update*
  else
    return $w = \min(\frac{|\delta_t|}{20}, 1)$, $t_c = t_{now}$
  end if
end

command-for-desired-location function generates the desired command to arrive at the goal point. First, considering the overlapping areas of the two images, we calculate p_{coll_l} and δ_{h_l} from the left image, and p_{coll_r} and δ_{h_r} from the right image. Then, we calculate the overall collision probability, p_{coll_t}. Assuming that the maximum speed is V_{max}, a velocity command is generated such that the velocity becomes zero when the collision probability is high. The next step is to generate the heading angle ψ_t. The target heading angle δ_g is easily calculated using the coordinates of the target and current positions. However, it is crucial to ensure that the effects of the target heading angle δ_g and the heading angle for obstacle avoidance δ_h do not cancel each other out. To do so, the *get-weight* function sets the weight w_{time} close to one when the avoidance angle is large, ensuring that only commands to avoid obstacles are executed. When the collision avoidance angle is small, w_{time} approaches zero, allowing the target angle command to be executed for following the target point. Additionally, the weight was set to change smoothly through the exponential function, $e^{-0.05 \cdot (t_{\text{now}} - t_c)}$. To analyze the performance of our algorithm, we performed a flight simulation to reach the target point, and detailed results will be discussed in Section VI-A.

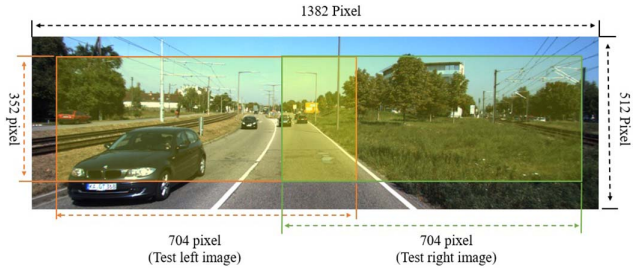


Fig. 7. Test image generation on KITTI dataset using our configuration.

V. Performance Analysis for Depth Estimation

A. Performance Comparison on Outdoor Datasets

Before conducting the tests, we made test datasets from the KITTI, Virtual KITTI2, ApolloScape, and DDAD datasets. Since our algorithm requires stereo depth estimation based on the small overlapping areas of images, we modified each dataset accordingly as shown in Fig. 7. The modified datasets enabled us to derive regression parameters (3), which were then utilized in our estimation algorithm. Our evaluation focused on both the left and right images from each image pair, allowing for a comprehensive evaluation. During the depth-refinement process, we inversely estimated the baseline between the image pair for stereo matching by exploiting the actual depth values provided by KITTI, etc.

The advantage of our depth estimation algorithm lies in its ability to show fast and accurate performance even within a lightweight network. Moreover, it exhibits exceptional estimation accuracy even in unseen environments, thanks to the depth-refinement technique employed. To verify this performance, we also conducted a comparison of the depth estimation performance by applying the algorithm to both the environment in which it was trained, utilizing the KITTI dataset, and the unseen dataset, the Virtual KITTI2, DDAD, and ApolloScape dataset. The results can be found in Table II and Fig. 8.

The accuracy δ_i, Absrel, and RMSE are calculated as

$$\%\text{ of } d_p \; s.t. \max\left(\frac{d_p}{d_p^*}, \frac{d_p^*}{d_p}\right) = \delta < \text{threshold} \tag{7}$$

where

$$\text{Absrel} = \frac{1}{N}\sum \frac{\|d_p - d_p^*\|}{d_p^*}, \text{RMSE} = \sqrt{\frac{1}{N}\sum (d_p - d_p^*)^2},$$

and d_p and d_p^* are the estimated and ground-truth depths for pixel p, respectively. N is the total number of data. In this study, we use a threshold as $\delta_1 = 1.25$, $\delta_2 = (1.25)^2$, and $\delta_3 = (1.25)^3$. In addition, because unfilled regions where depth information is missing exist in a depth image, we do not compute the accuracy, δ, if $d_p^* = 0$. Fig. 8 shows the estimated depth images in an outdoor scenario. When comparing on KITTI (i.e, trained:KITTI, test:KITTI), or vKITTI2 (i.e, trained:KITTI, test:vKITTI2) our

TABLE II
QUANTITATIVE RESULTS TO SEEN OR UNSEEN DATASETS

Models	Datasets	Resolution	Inference Time (ms)	KITTI (Seen)			vKITTI2 (Unseen)			ApolloScape (Unseen)			DDAD (Unseen)		
				δ_1	Absrel	RMSE	δ_1	Absrel	RMSE	δ_1	Absrel	RMSE	δ_1	Absrel	RMSE
Monodepth2 [10]	K	640*192	**9.89**	0.931	0.078	3.2444	0.835	0.121	5.515	0.089	0.4895	5.184	0.768	0.160	8.550
GLPDepth [15]	K	704*352	75.80	0.967	0.057	2.297	0.884	0.103	4.380	0.063	0.468	5.272	0.827	0.139	7.022
AdaBins [16]	K	704*352	90.963	0.964	0.058	2.360	0.853	0.106	4.756	0.074	0.469	5.448	0.752	0.169	8.914
NewCRFs [11]	K	704*352	181.58	**0.975**	**0.052**	2.072	0.848	0.109	5.359	0.075	0.459	5.277	**0.843**	**0.119**	**6.177**
ZoeDepth [17]	M+NK*1	704*352	96.98	0.966	0.057	2.362	0.850	0.102	5.055	0.074	0.475	4.473	0.824	0.169	8.914
Ours	K	704*352	21.66	0.959	0.065	2.436	0.860	0.116	4.766	0.079	0.451	4.874	0.790	0.148	7.918
Ours with Depth refinement	K	704*352	49.11 *2	0.973	0.056	**2.030**	**0.910**	**0.095**	**3.977**	**0.797**	**0.160**	**2.977**	**0.843**	**0.119**	6.779

Note: The best result is in bold, second best is underlined. *1 : Midas+NYU+KITTI, *2 : Ours with refinement measured the time per single image.

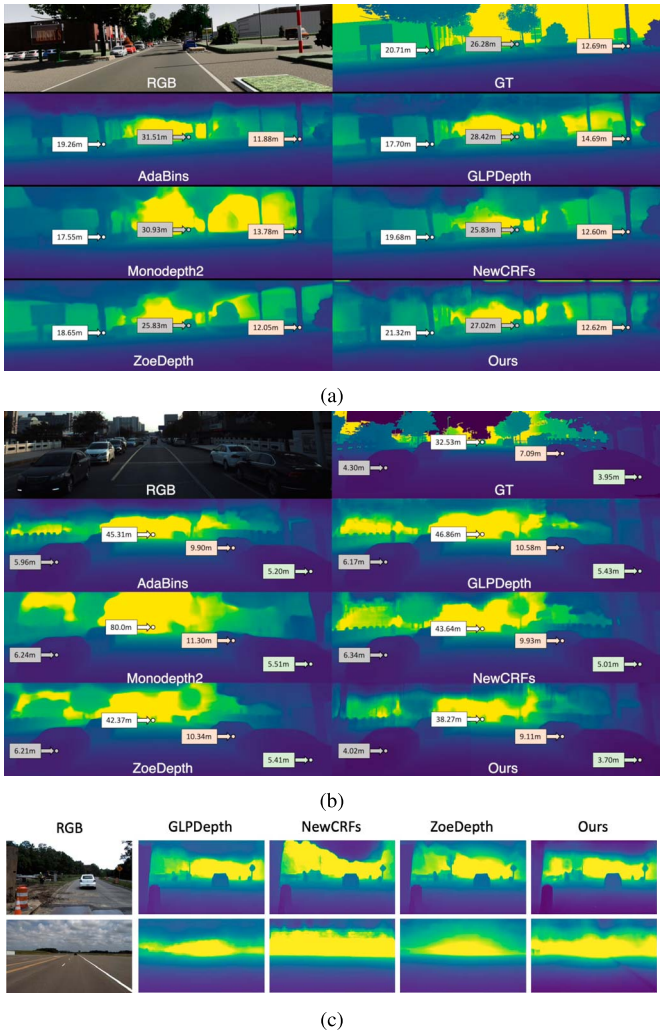


Fig. 8. Depth estimation performance in outdoor environment. (a) Test depth image on the virtual KITTI2 dataset. (b) Test depth image on the Apollo dataset. (c) Test depth image on DDAD dataset.

algorithm demonstrated the second-best or the best depth estimation performance, respectively. NewCRFs showed the best performance in KITTI, but it is very slow for practical applications. This outstanding result was achieved with rapid and

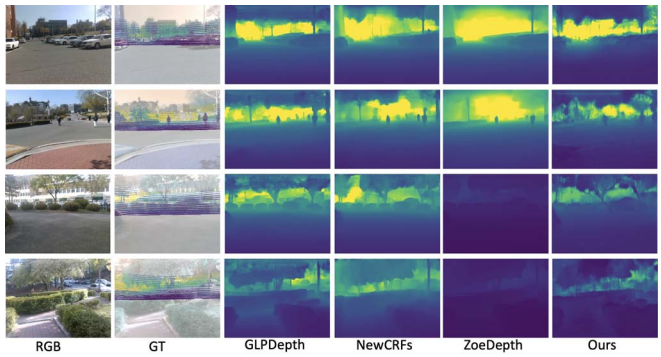


Fig. 9. Test depth on our collected dataset.

precise inference speed and a lightweight network. Additionally, when applied to ApolloScape (i.e, trained:KITTI, test: ApolloScape), an unseen environment, our algorithm exhibited exceptional performance compared to other algorithms. Even on the DDAD dataset, we demonstrated the most accurate estimation performance. The key factor contributing to this success was the depth refinement algorithm.

B. Performance Comparison on Our Dataset

Our outdoor test data are collected using two monocular cameras and Livox LiDAR. The FOV of a monocular camera is 110-degree, so our stereo setup is 160 degrees. The FOV of Livox LiDAR is 120 degrees. To obtain the RGB-aligned depth, the extrinsic parameters are estimated using camera-LIDAR calibration [30].

Fig. 9 shows the estimated depth images in an outdoor scenario. We compared our proposed algorithm with state-of-the-art outdoor depth estimation algorithms including AdaBins [16], GLPdepth [15], Monodepth2 [10], Zoedepth [17], and NewCRFs [11]. The performance table is summarized in Table III. Here, our proposed algorithm outperforms other state-of-art depth estimation algorithms. Our algorithm exhibited significantly higher depth estimation performance, particularly in the range of 15 m or less.

We also analyzed depth estimation from images acquired from a car traveling at 50 km/h to show the advantages of the MDE algorithm. As shown in Fig. 10, although the Monodepth2

TABLE III
δ_1 QUANTITATIVE RESULTS IN FIG. 9 AND ESTIMATION PERFORMANCE COMPARISON USING OUR DATASETS

Models	Outdoor Accuracy (32 images, ≤80 m)						
	image1	image2	image3	image4	δ_1	Absrel	RMSE
MonoDepth2 [10]	**0.465**	0.359	0.121	0.276	0.262	0.673	16.495
GLPDepth [15]	0.410	0.188	0.086	0.201	0.320	0.537	16.286
AdaBins [16]	0.231	0.095	0.043	0.059	0.131	1.849	26.518
NewCRFs [11]	0.406	0.342	0.095	0.254	0.298	0.664	14.027
ZoeDepth [17]	0.421	0.128	0.000	0.000	0.298	0.531	**13.667**
Ours	0.430	**0.661**	**0.697**	**0.390**	**0.340**	**0.422**	15.750
Models	Outdoor Accuracy (32 images, ≤15 m)						
	image1	image2	image3	image4	δ_1	Absrel	RMSE
MonoDepth2 [10]	0.307	0.122	0.007	0.073	0.157	1.044	9.589
GLPDepth [15]	0.262	0	0.002	0.035	0.176	0.793	7.639
AdaBins [16]	0	0	0	0	0.002	3.213	28.327
NewCRFs [11]	0.032	0.001	0	0.004	0.108	1.062	9.029
ZoeDepth [17]	0.002	0	0	0	0.040	0.912	9.120
Ours	**0.563**	**0.950**	**0.895**	**0.695**	**0.645**	**0.537**	**6.258**

Note: The bold values indicates the best results.

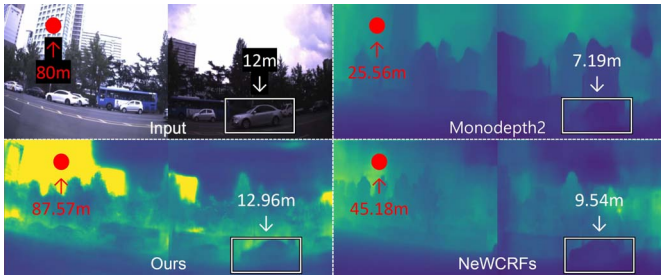


Fig. 10. Depth estimation performance in outdoor environment (vehicle speed about 50 km/h).

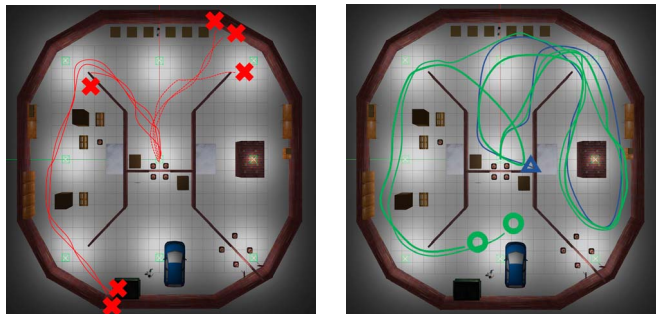


Fig. 11. Trajectory in simulated flights with a top-down view of the world (25 m × 25 m). (a) Flights with DroNet [6] (solid red line) and Chakravarty et al. [5] (dotted red line). (b) Flight with the proposed method (success case: solid green line, failure case: solid blue line).

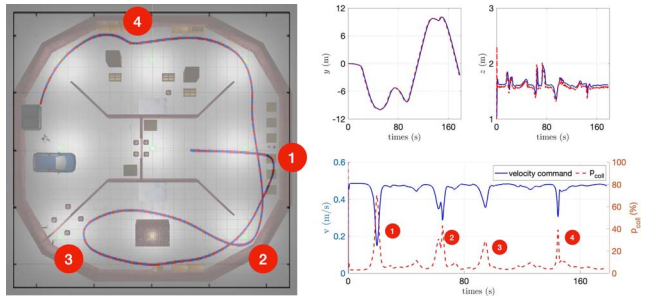


Fig. 12. Our flight with 3-D obstacles. (a) 3-D flight in 25 m × 25 m world. (b) Collision probability with velocity command.

performed the quickest estimation, the boundary line of the object was not properly estimated, and it worsens with the increase in the speed of the quadrotor or the moving obstacle. The major advantage of our proposed algorithm is its superior accuracy compared to other benchmark algorithms [10], [11], especially when the vehicle or obstacle speed is relatively high. This is because our algorithm demonstrates fast and precise estimation compared to the other benchmark algorithms.

VI. EXPERIMENTS FOR COLLISION AVOIDANCE

The performance of the proposed algorithm was verified by comparing it with other algorithms [5], [6] in a simulation environment. In addition, we conducted real experiments using our custom-made wide-FOV stereo camera.

A. Simulation Results

For the comparison, we designed the gazebo environment (Fig. 11). A wide stereo camera was designed, with each camera having an 82-degrees horizontal FOV. The total horizontal FOV of our stereo camera was 112-degrees.

The avoidance results are shown in Fig. 11. Fig. 11(a) shows the obstacle avoidance results of DroNet [6] and a monocular-based obstacle avoidance [5]. Because the deep learning network of DroNet [6] was trained with real outdoor images, its performance might degrade owing to the differences between the real images taken outdoors and the virtual images obtained from the simulations. The method in [5] also leads to collisions with obstacles in complex environments because it is developed based on a lower depth estimation performance than the state-of-the-art depth estimation [16], [24]. Moreover, its collision probability is high owing to the limited FOV of a monocular camera. Fig. 11(b) shows our exploration results. Our algorithm shows the best avoidance performance among all compared methods [5], [6]. This is because more accurate depth estimation is possible by exploiting the reliable depth obtained from the stereo matching, although the proposed algorithm also uses a CNN learned from real images. In addition, the algorithm in [5] cannot avoid 3-D obstacles, our proposed method can avoid 3-D obstacles as shown in Fig. 12(a) and 12(b). Our algorithm avoids 3-D collisions satisfactorily.

Fig. 13 shows the simulation results of flying to the target point using Algorithm 1. The simulations were conducted in a 100 m × 100 m forest-simulated gazebo environment. Depth refinement and avoidance commands were generated using images from two monocular cameras. Depth estimation was performed using a model trained on the KITTI dataset. Despite the simulation environment differing from the training dataset, our algorithm achieved satisfactory estimation performance [Fig. 13(c)]. We confirmed that the proposed algorithm effectively reached the destination without the need for complex mapping or complex path planning algorithms such as RRT or optimization-based planning [8].

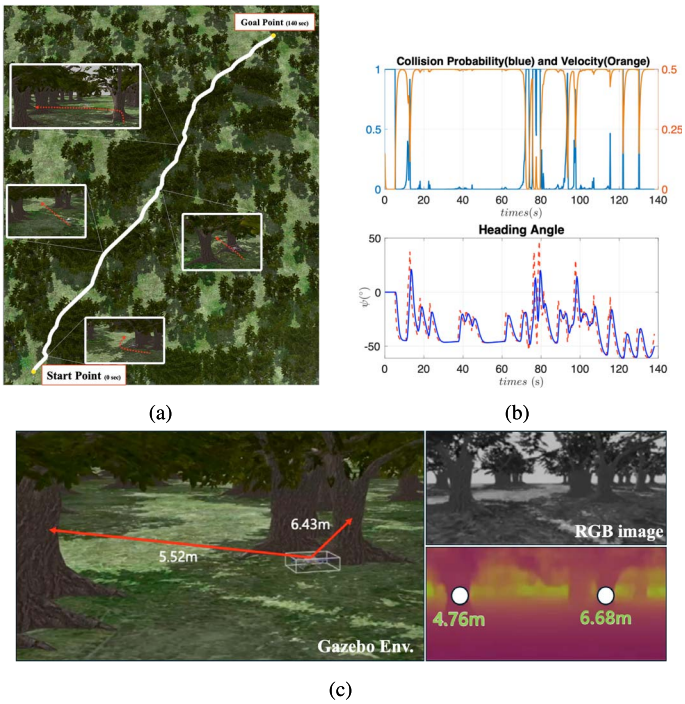


Fig. 13. Simulation result in the forest environment (100 m $\times$ 100 m). (a) Flight results. (b) Desired velocity v_d and heading angle ψ_t. and (c) Snapshot during simulation.

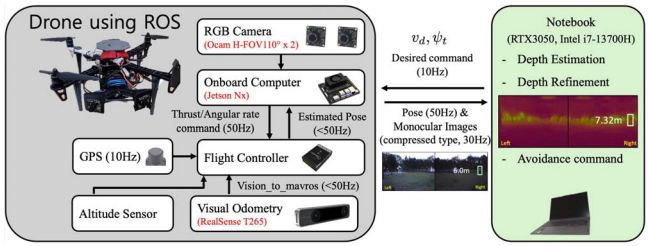


Fig. 14. Experimental setup.

B. Experimental Results

Based on the proposed algorithm, we conducted experiments with an obstacle in an outdoor environment. A quadrotor equipped with an onboard computer (Jetson NX) and two monocular cameras as shown in Fig. 14. Each monocular camera used in the experiments has a horizontal FOV of 110 degrees. The total horizontal FOV of our stereo camera is 160 degrees. Thanks to our real-time depth estimation algorithm, avoidance commands can be generated at a rate exceeding 10 Hz.

Fig. 15 shows the outdoor experiment for avoiding dynamic obstacles. The depth estimation performance was satisfactory even for a person in motion. However, there was a slight error in depth estimation, but it was small and did not affect obstacle avoidance. The outdoor experiment, using depth estimation learned from an outdoor dataset, demonstrates that the proposed algorithm with depth estimation and obstacle avoidance shows

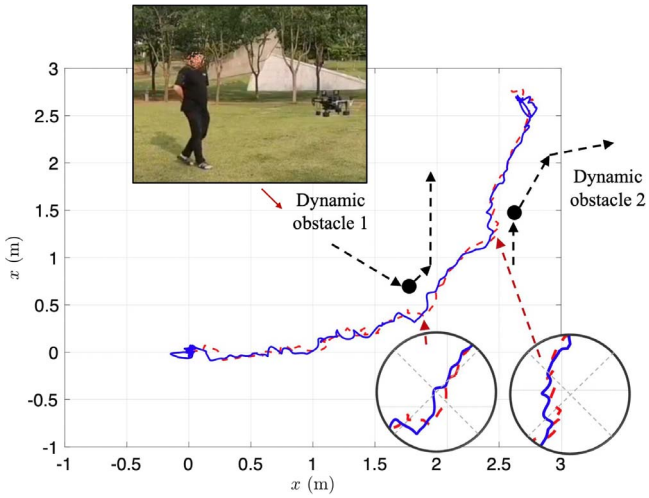


Fig. 15. Picture taken during the outdoor experiment for a dynamic obstacle.

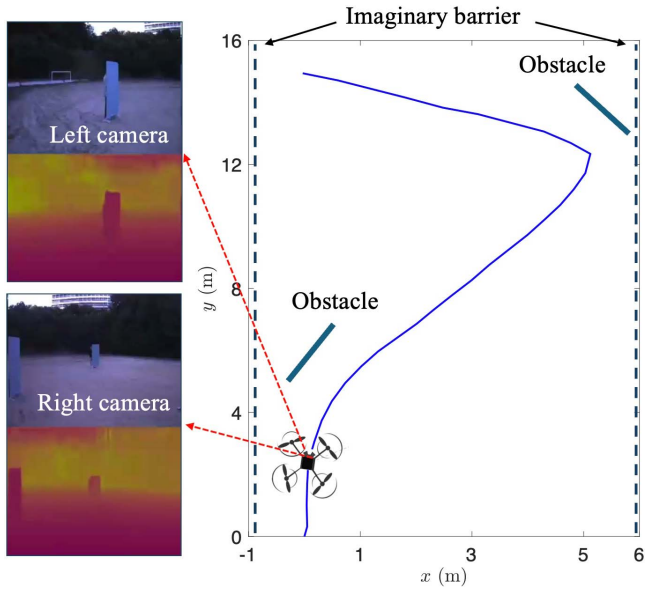


Fig. 16. Trajectory of the quadrotor in the outdoor experiment for a static obstacle.

satisfactory results even outdoors. Fig. 16 shows the outdoor experiment for avoiding static obstacles. In the experiment, we created an environment with virtual walls at -1 m and 6 m and included two static obstacles. The experimental results demonstrate that the UAV efficiently navigates through the environment while avoiding the two obstacles.

VII. Conclusion

In this study, we have proposed an obstacle avoidance algorithm for micro air vehicles (MAVs) by estimating the depth using a wide-FOV stereo camera. We exploited a MDE algorithm and improved the estimation accuracy by using a noise removal filter with a superpixel technique. A reliable depth value,

calculated by stereo-matching in some overlapping regions of two images, was used to compute the real-world scale. We developed a simple-to-calculate avoidance algorithm to generate the target speed and heading angle of an MAV based on the estimated depth. We conducted simulations and experiments and verified that our algorithm ensured safety and computational efficiency for obstacle avoidance by an MAV.

In our future work to enhance the ability of UAVs, we plan to focus on two key areas: 1) expanding depth estimation capabilities for broader environments; and 2) enhancing depth estimation performance in extreme weather conditions. To achieve this goal, we intend to utilize a small number of fisheye cameras to develop recognition technology for wider environments, enabling 360-degree depth perception. Additionally, we aim to advance depth estimation technology specifically tailored for extreme weather environments, such as severe fog.

References

[1] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *J. Field Robot.*, vol. 37, no. 8, pp. 1363–1388, 2020.

[2] P. K. Muthusamy, M. Garratt, H. Pota, and R. Muthusamy, "Real-time adaptive intelligent control system for quadcopter unmanned aerial vehicles with payload uncertainties," *IEEE Trans. Ind. Electron.*, vol. 69, no. 2, pp. 1641–1653, Feb. 2022.

[3] H. Lee, H. Seo, and H.-G. Kim, "Trajectory optimization and replanning framework for a micro air vehicle in cluttered environments," *IEEE Access*, vol. 8, pp. 135406–135415, 2020.

[4] M. G. Mueller et al., "Robust visual-inertial state estimation with multiple odometries and efficient mapping on an MAV with ultra-wide fov stereo vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 3701–3708.

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

[6] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[7] X. Dai, Y. Mao, T. Huang, N. Qin, D. Huang, and Y. Li, "Automatic obstacle avoidance of quadrotor UAV via CNN-based learning," *Neurocomputing*, vol. 402, pp. 346–358, Aug. 2020.

[8] Y. Lin et al., "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 35, no. 1, pp. 23–51, 2018.

[9] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Sci. Robot.*, vol. 5, no. 40, 2020, Art. no. eaaz9712.

[10] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3828–3838.

[11] W. Yuan, X. Gu, Z. Dai, S. Zhu, and P. Tan, "NewCRFs: Neural window fully-connected CRFs for monocular depth estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3906–3915.

[12] J. Lee et al., "SlaBins: Fisheye depth estimation using slanted bins on road environments," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8765–8774.

[13] U. Shin, J. Park, and I. S. Kweon, "Deep depth estimation from thermal image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 1043–1053.

[14] K. Wang, W. Ding, and S. Shen, "Quadtree-accelerated real-time monocular dense mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 1–9.

[15] D. Kim, W. Ka, P. Ahn, D. Joo, S. Chun, and J. Kim, "Global-local path networks for monocular depth estimation with vertical cutdepth," 2022, *arXiv:2201.07436*.

[16] S. F. Bhat, I. Alhashim, and P. Wonka, "AdaBins: Depth estimation using adaptive bins," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4009–4018.

[17] V. Guizilini, I. Vasiljevic, D. Chen, R. Ambrus, and A. Gaidon, "Towards zero-shot scale-aware monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 9233–9243.

[18] G. Cho, J. Kim, and H. Oh, "Vision-based obstacle avoidance strategies for MAVs using optical flows in 3-d textured environments," *Sensors*, vol. 19, no. 11, 2019, Art. no. 2523.

[19] K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 1070–1076, Apr. 2017.

[20] N. J. Sanket, C. D. Singh, V. Asthana, C. Fermüller, and Y. Aloimonos, "MorphEyes: Variable baseline stereo for quadrotor navigation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2021, pp. 413–419.

[21] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 2366–2374.

[22] C. Won, J. Ryu, and J. Lim, "End-to-end learning for omnidirectional stereo matching with uncertainty prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 3850–3862, Nov. 2021.

[23] Y. Yang, M. Pan, S. Jiang, J. Wang, W. Wang, J. Wang, and M. Wang, "Towards autonomous parking using vision-only sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS),* Piscataway, NJ, USA: IEEE Press, 2021, pp. 2038–2044.

[24] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," 2019, *arXiv:1907.10326*.

[25] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp.1623–1637, Mar. 2022.

[26] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, "ConvNeXt V2: Co-designing and scaling convnets with masked autoencoders," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16133–16142.

[27] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," 2014, *arXiv:1403.1687*.

[28] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.

[29] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, 2010, pp. 25–38.

[30] Livox, "Livox camera lidar calibration," GitHub. [Online]. Available: https://github.com/Livox-SDK/livox_camera_lidar_calibration

Euihyeon Cho received the B.S. and M.S. degrees in electrical engineering from Kyungpook National University, Daegu, South Korea, in 2023 and 2024, respectively.

His research interests include aerial robots and learning-based path planning of aerial robots.

Hyeongjin Kim received the B.S. and M.S. degrees in electrical engineering from Kyungpook National University, Daegu, South Korea, in 2020 and 2022, respectively.

His research interests include deep learning and learning-based path planning of aerial robots.

Pyojin Kim received the B.S. degree in mechanical engineering from Yonsei University, in 2013, and the M.S. and Ph.D. degrees in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2015 and 2019, respectively.

He is an Assistant Professor with the School of Mechanical and Robotics Engineering, Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea. His research interest includes computer vision.

Hyeonbeom Lee received the B.S. degree in mechanical and control engineering from Handong Global University, Pohang, South Korea, in 2011, and the M.S. and Ph.D. degrees in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2013 and 2017, respectively.

He is an Associate Professor with the Department of Mechanical Engineering, Ajou University, Suwon, South Korea. His research interest includes autonomous navigation of aerial robots.