

Article

Pix2Pix-Based Monocular Depth Estimation for Drones with Optical Flow on AirSim

Tomoyasu Shimada ¹, Hiroki Nishikawa ^{1,2}, Xiangbo Kong ^{1,*} and Hiroyuki Tomiyama ¹

¹ Graduate School of Science and Engineering, Ritsumeikan University, Kusatsu 525-8577, Japan ; ri0080xs@ed.ritsumei.ac.jp (T.S.); hiroki.nishikawa@tomiyama-lab.org (H.N.); ht@fc.ritsumei.ac.jp (H.T.)

² Japan Society for the Promotion of Science, Tokyo 102-0083, Japan

* Correspondence: kong@fc.ritsumei.ac.jp; Tel.: +81-77-561-5013

Abstract: In this work, we propose a method for estimating depth for an image of a monocular camera in order to avoid a collision for the autonomous flight of a drone. The highest flight speed of a drone is generally approximate 22.2 m/s, and long-distant depth information is crucial for autonomous flights since if the long-distance information is not available, the drone flying at high speeds is prone to collisions. However, long-range, measurable depth cameras are too heavy to be equipped on a drone. This work applies Pix2Pix, which is a kind of Conditional Generative Adversarial Nets (CGAN). Pix2Pix generates depth images from a monocular camera. Additionally, this work applies optical flow to enhance the accuracy of depth estimation. In this work, we propose a highly accurate depth estimation method that effectively embeds an optical flow map into a monocular image. The models are trained with taking advantage of AirSim, which is one of the flight simulators. AirSim can take both monocular and depth images over a hundred meter in the virtual environment, and our model generates a depth image that provides the long-distance information than images captured by a common depth camera. We evaluate accuracy and error of our proposed method using test images in AirSim. In addition, the proposed method is utilized for flight simulation to evaluate the effectiveness to collision avoidance. As a result, our proposed method is higher accuracy and lower error than a state of work. Moreover, our proposed method is lower collision than a state of work.

Keywords: depth estimation; optical flow; AirSim



Citation: Shimada, T.; Nishikawa, H.; Kong, X.; Tomiyama, H.

Pix2Pix-Based Monocular Depth Estimation for Drones with Optical Flow on AirSim. *Sensors* **2022**, *22*, 2097. <https://doi.org/10.3390/s22062097>

Academic Editor: Loris Nanni

Received: 13 January 2022

Accepted: 4 March 2022

Published: 8 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, small drones have been more popular than ever from the perspective of flexibility, low power consumption, and reasonable prices. In addition, the drones are expected to play a variety of roles to take advantage of their convenience. The roles include infrastructure inspection, package delivery, and mobile surveillance cameras. Unlike manned vehicles such as cars and airliners, unmanned drones do not need to be controlled by a person and autonomous flights are becoming practical. In terms of autonomous flights of drones, collision avoidance has been indispensable and regarded as one of the crucial issues. Typically, conventional solutions have employed distance sensors. For instance, Light Detection and Ranging (LiDAR) which can detect long distances are employed [1,2]. Depth cameras or stereo cameras are also employed to perceive distance [3–6]. However, such sensors with high performance are usually heavy, costly, and power-consuming to equip on a small drone. In contrast, low performance depth sensors can hardly have long-distance vision with high accuracy and would rather increase risk of collisions with objects.

Many kinds of research for autonomous flight of drones have assumed that monocular cameras are often used to detect and recognize objects around the drones [7,8]. Single monocular camera-based depth estimation is also actively researched [9–12]. However, monocular cameras are not useful at night in terms of their visibility. Instead, infrared cameras are employed to improve the visibility at night, but they do not include depth

information, which means the distances between a drone and objects, as well as monocular images. In the literature, depth images have been more important than ever to measure how far the objects are placed from the drone. A lightweight depth camera, which is small enough to be mounted on a drone, can measure a distance up to only 10 m. In addition, unfortunately, high-performance depth cameras are often expensive and too heavy to be equip on a drone. Therefore, depth estimation technology from an image by monocular cameras has been extensively investigated.

The contributions of this paper are threefold as follows:

- This is the first paper to generate a depth image from a monocular image with optical flow for collision avoidance of drone flight.
- We verify that our proposed method can estimate high-quality depth images in real-time, and demonstrates that a drone can successfully fly avoiding objects in a flight simulator.
- In addition, our method is superior to previous method of depth estimation on accuracy and collision avoidance.

The rest of this paper is organized as follows. Related work of drone autonomous flight method and depth estimation method is introduced in Section 2. Section 3 describes the overview of AirSim. Section 4 shows a proposed method to estimate depth with optical flow. Section 5 shows the experimental results and Section 6 concludes this paper.

2. Related Work

There has been a great deal of work related to autonomous drone flying for several decades. Much of work has been focused on safe flight, which is particularly necessary to prevent collision with objects. These studies include obstacle avoidance based on ultrasonic, radar, and image processing [13]. Ultrasonic-based methods perform in real-time but the maximum range is short [14,15]. Radar-based methods perform well in obstacle detection. However, radar is not a good choice for small Unmanned Aerial Vehicles (UAVs) due to its weight [16]. Vision-based methods include obstacle avoidance methods based on LiDAR images, Time of Flight (ToF) images, binocular images, or monocular images. In [1,2,17], the authors used LiDAR for collision avoidance of a drone. However, the installation of many powerful sensors or high performance sensors results in an increase in the weight of a drone, which leads to an increase in energy consumption. Given that the drone's flight is limited by the battery capacity, it is difficult for the drone to fly long distances with a large number of sensors.

In order to tackle this issue, the approaches of [3,4,6] proposed collision avoidance techniques using a light and small depth camera and a stereo camera. The presented methods enable a drone to avoid obstacles on-the-fly by determining an optimum flight direction using depth images. The work in [4], which is inspired by [3,6], proposed an algorithm for collision avoidance by dividing an image from a depth camera into five sections and selects a section so that the section is the most distant object among them. However, depth cameras with reasonable price and relatively low weight such as Kinect released by Microsoft can hardly be mounted on a drone since they can measure only within 10 m [18]. In the context, depth estimation from a monocular camera, which can overlook farther than a depth camera, has been attractive.

In the studies [9,10], the authors presented the methods of depth estimation using Support Vector Machine (SVM). The systems divide an image from a drone into patches, represent each patch using a set of manually created features, and estimate the depth of each patch using a pre-trained SVM classifier. However, the accuracy of their systems is not high as a result of handheld training data. In [19–22], these methods are based on Convolutional Neural Networks (CNN). The CNN-based methods are more accurate than the SVM method in [9,10], but the accuracy is still not sufficient enough to realize secure flight without collisions. In [11,23], the authors proposed methods to preprocess segmenting images before depth estimation using a monocular camera. This method improves the accuracy of depth estimation. On the other hand, the computational workload

largely increases, and the method is not suitable for real-time processing in terms of the performance since the methods [11,23] need segmentation as pre-process. In addition, the methods [11,19–23] use public datasets [24,25] to estimate depth. As the latest work, AdaBins based on a transformer has been proposed [26]. However, the data that the works have used for training and testing in [24,25] are unsuitable for drones view since the data in [24] is suitable for ground vehicles and the data in [25] is oriented to indoor environments. In [12], the authors collected data from drone views in a drone flight simulator and presented a method to generate depth images using Pix2Pix [27] from a monocular image.

Drones are further smaller than general vehicles, and the processing capability of drones is comparatively lower than that of cars since the large computer cannot be equipped on the drone. Most of recent technologies based on image processing have exhausted computational resources due to the development of deep neural networks (DNNs), and the technologies are seemingly not suitable to the system in a small device. However, many embedded systems are oriented to the Internet of things (IoT), and the computation with data transmission to an on-land computer has been enabling to distribute the computational workloads. Although this fashion has spurred the development of depth estimation technologies using a monocular camera, there is little work that focuses on depth estimation for a small drone. Drones are required to fly without colliding objects but the weight of a camera that can be carried by a drone is severely limited, and high performance but heavy cameras cannot be carried.

In this paper, we propose a new depth estimation method for autonomous flight of a drone. Our proposed method can estimate long distance using a monocular image with optical flow. In addition, our model for the estimation is based on conditional generative adversarial networks (CGAN) [28], and the training dataset is collected from AirSim [29], which is known as the virtual flight environment of a drone.

3. AirSim

This section describes AirSim [29], which we employ in this work. AirSim is a kind of flight simulator that uses a virtual environment called Unreal Engine 4. This simulator faithfully reproduces the reality in visual information and physics.

In addition, AirSim can acquire mesh information from Unreal Engine 4. The available information includes location, temperature, and images. The obtained images include RGB, segmentation, infrared, and depth images. The depth image in AirSim can exactly measure up to 200 m. Therefore, the AirSim environment enables to obtain and label both RGB and accurate depth images at the same time to create training dataset. Figure 1a shows a depth image up to 10 m taken by a real depth camera that can be installed on real small drones in the AirSim environment. Figure 1b shows a depth image up to 200 m obtained from a depth camera in AirSim. Figure 1c shows monocular image taken by monocular camera in AirSim. As shown in Figure 1, in monocular image and depth image up to 200 m, can detect objects, however depth image up to 10 m cannot detect that. Therefore, the more distance can be measured, the larger the benefit of using depth images for autonomous flight. However, to obtain a 200 m deep image with a real drone, it is necessary to install a camera of great depth, which is unrealistic. Therefore we propose a method for estimating the depth image obtained by AirSim based on a monocular image.

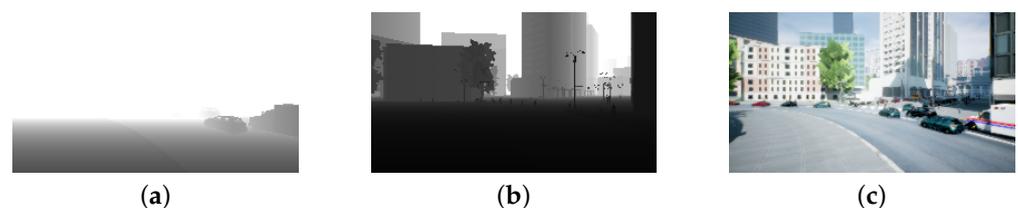


Figure 1. AirSim views: (a) Depth map (up to 10 m), (b) Depth map (up to 200 m), (c) Monocular (RGB) image of (b).

4. A Pix2Pix-Based Monocular Depth Estimation with Optical Flow

This section describes a proposed depth estimation method, which is based on Pix2Pix [27]. Figure 2 shows the system overview of our proposed method. Here, we briefly address our proposed method. The proposed method consists of three parts: The first part generates an optical flow map from two adjacent frames. Second, we combine the generated optical flow map with a monocular image. Finally, the combined image is input into a Pix2Pix-based depth estimator to create a depth image. In the following, we detail each part of the proposed method.

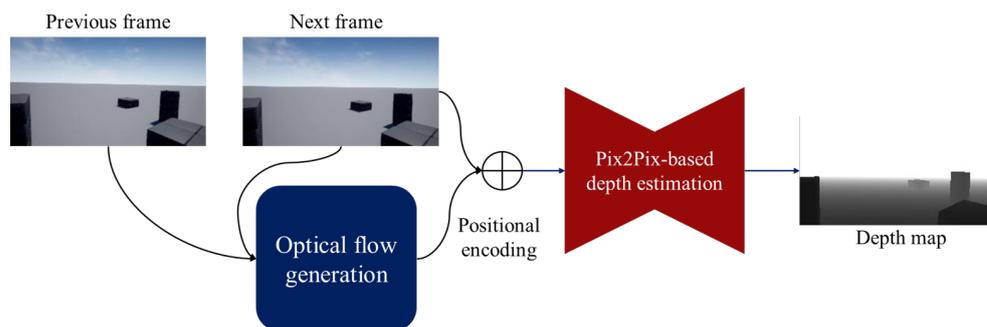


Figure 2. System overview of the proposed method.

4.1. Optical Flow Map Generation

We employ classical methods to generate an optical flow map based on Lucas-Kanade [30] and Farneback method [31]. First, we describe the image feature representation and its spatio-temporal analysis in Lucas-Kanade method. Lucas-Kanade method assumes that the deformation of an object between two adjacent frames is slight and that a point x on image f_t at time t has moved by v_t on image f_{t+1} at time $t+1$.

$$\epsilon(v_t) = \sum_{d \in B} (f_{t+1}(x + v_t + d) - f_t(x + d))^2 \quad (1)$$

Here, B represents a certain rectangular window region centered at point x , and d is a parameter that represents an arbitrary position in the window region. It is the sum of the squares of the luminance differences between the corresponding points in the frame area before and after the movement of point x , where v_t is the movement from time t to $t+1$. For example, if all the points in the frame region have the same luminance and have moved in the same direction by the same amount, then Equation (1) becomes 0. Approximating the right-hand side of Equation (1) with a first-order Taylor expansion yields the following Equation (2).

$$\epsilon(v_t) = \sum_{d \in B} (\nabla f_t^T(v_t) - \dot{f}_t)^2 \quad (\nabla f = \frac{\partial f}{\partial x}, \quad \dot{f}_t = \partial f_t = \partial t) \quad (2)$$

Here, ∇f_t represents the horizontal and vertical difference values of the image f_t at point x . \dot{f}_t represents the time difference value between adjacent frames at point x . Lucas-Kanade method [30] is used to find the displacement v at each position x that minimizes the sum of the squared luminance differences. By differentiating Equation (2) by v_t and setting it to 0, the optimal travel distance v_t is obtained as follows:

$$v_t = -G^{-1}b \quad (3)$$

$$G = \sum_{d \in B} \nabla f_t^T (\nabla f_t)^T \quad (4)$$

$$b = \sum_{d \in B} \dot{f}_t \nabla f_t \quad (5)$$

Since the amount of movement at each point is not independent, the following iterative process is performed until v_t^{k+1} does not change at all points x , and the amount of movement is determined.

$$v_t^{k+1}(x) = v_t^k(x) - Gb \quad (6)$$

On the other hand, Farneback method [31] approximates the luminance value of each pixel with a second-order polynomial, and estimates the amount of movement with high accuracy by comparing the coefficients between frames. Let $f_t(x) \in [0, 1]$ denote the luminance value of coordinate x at time t . The luminance values in the neighborhood of x are expressed as second-order polynomials, and the coefficients are optimized by the weighted least-squares method in Equation (7).

$$\hat{f}_t(x) = x^T A_t x + b_t^T x + C_t \quad (7)$$

A_t, b_t, c_t are a (2,2) symmetric matrix, a (2,1) column vector, and a scalar, respectively. Let v_t denote the movement of point x at time t until time $t + 1$. From $\hat{f}_t(x) = \hat{f}_{t+1}(x + v_t)$, the movement v_t can be estimated as Equation (8).

$$v_t = -\frac{1}{2} A_t^{-1} (b_{t+1} - b_t) \quad (8)$$

In order to obtain a stable solution, Farneback method approximates the coefficient A_t as follows Equation (9).

$$\hat{A}_t = \frac{A_t + A_{t+1}}{2} \quad (9)$$

Then, using \hat{A}_t instead of A_t in Equation (8), we obtain Equation (10).

$$\hat{A}_t v_t = \Delta b_t \quad (10)$$

Equation (10) holds for all points x . Farneback method also considers the neighborhood around a point x , and introduces the following energy function.

$$\epsilon(v_t) = \sum_{d \in B} w(d) \|\hat{A}_t(x+d)v_t(x) - \Delta b_t(x+d)\|^2 \quad (11)$$

To minimize this energy, determine the ideal displacement $v_t(x)$ at point x is determined to minimize this energy. Farneback method is the same as Lucas-Kanade method. Farneback method is similar to Lucas-Kanade method, and is obtained by differentiating Equation (11) by $v_t(x)$ in the following:

$$v_t(x) = G^{-1}h \quad (12)$$

$$G = \sum_{d \in B} w(d) \hat{A}_t^T(x+d) \hat{A}_t(x+d) \quad (13)$$

$$h = \sum_{d \in B} w(d) \|\hat{A}_t^T(x+d) \Delta b_t(x+d)\|^2 \quad (14)$$

The actual displacement is estimated by iterative operation based on the above equation as in Lucas-Kanade method. Farneback method can obtain the concentration gradient stably by approximating the local image surface with a quadratic surface. In general, Farneback method provides more accurate tracking than Lucas-Kanade method, although the computational cost increases. Figure 3 shows inputs and an optical flow map using Farneback method [31]. As shown in Figure 3, the luminance value of near objects in the optical flow map is high. This figure indicates that relative motion of the objects near

a drone becomes large, while that of the objects far away from a drone becomes small. Hereby, we obtain the optical flow map in this way.

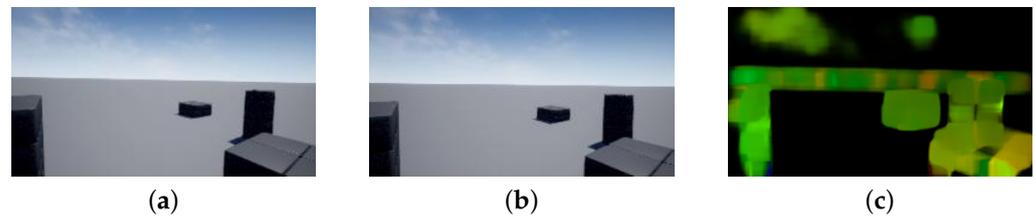


Figure 3. Optical flow map generated from inputs and outputs: (a) Previous frame, (b) Next frame, (c) Optical flow map.

4.2. Pix2Pix

In this work, our proposed method is based on Pix2Pix to generate a depth image from an monocular image [27]. Pix2Pix is well known method similar to CGAN [28]. Figure 4 is the overview of Pix2Pix, which represents the broad structure of the CGAN model. CGAN is typically split into two networks such as a generator and a discriminator. The generator learns to prevent the generated image from being detected by the discriminator as the generated one. The discriminator learns not to misidentify the training data and the generated data, and finally the generator is improved by the discriminator and can generates an image similar to the training data. The generator uses U-Net [32], which can extract local features and recover location information, and we show the concept of U-Net in Figure 5. The convolutional layer can extract local features as the layers get deeper. However, at the same time, the location information becomes ambiguous. Therefore, as shown in Figure 5, by sending the location information to the decoder side of the same layer, it is possible to extract local features and recover the location information.

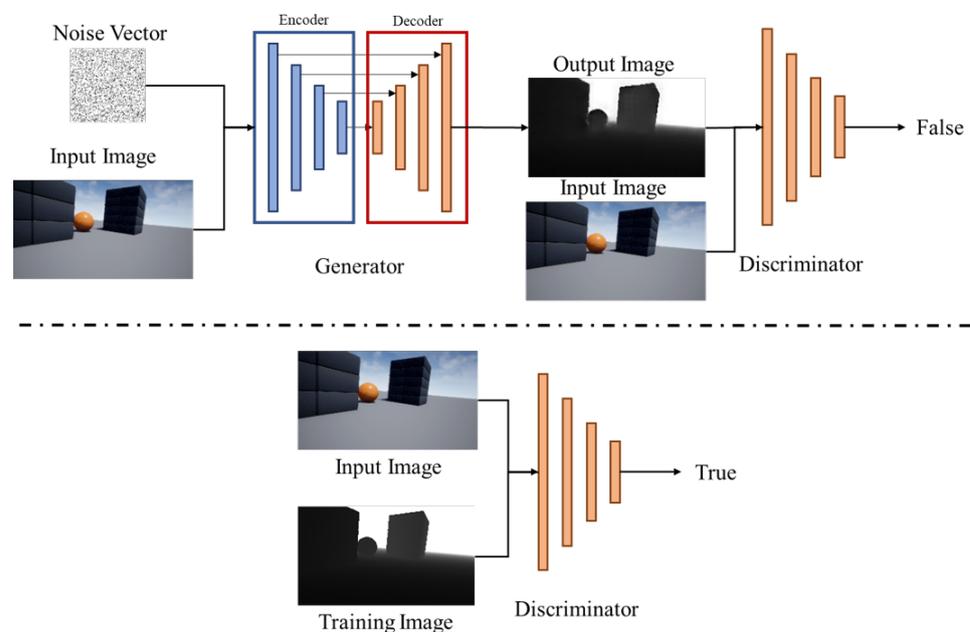


Figure 4. CGAN Network of Pix2Pix Architecture.

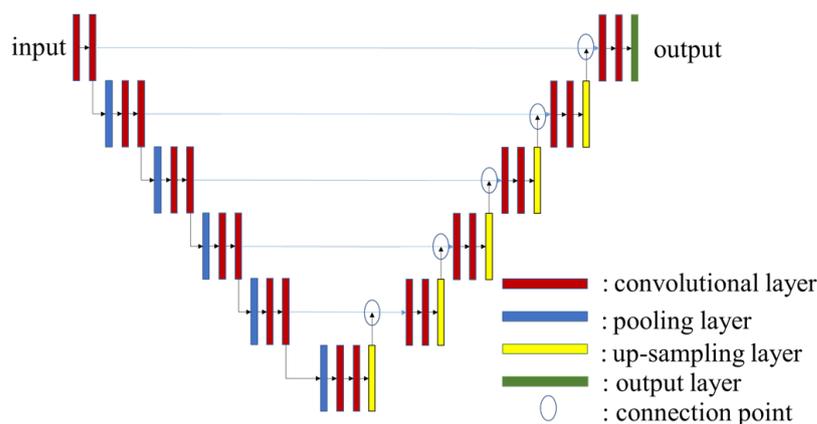


Figure 5. U-Net Architecture [32].

The objective of the CGAN that we have employed is as shown in the following equation, which is referred to [27].

$$\mathcal{L}_{CGAN}(G, D) = \mathbb{E}_{i,gt}[\log D(i, gt)] + \mathbb{E}_{i,n}[\log(1 - D(i, G(i, n)))] \quad (15)$$

Here, i is an input image and gt is ground truth. $D(i, gt)$ is the probability of judging the training data as training data, and $D(i, G(i, n))$ is the probability of judging the generated image as training data. Let $G(i, n)$ denote the generated image and n be a noise vector. The noise vector n is not necessary, but if training without n input, it results in poor flourishing performance. Therefore, this paper assumes to require the input of the noise vector. The discriminator tries to maximize this objective, while the generator tries to minimize it, and the generator needs to generate images that not only fool the discriminator but also come closer to the ground truth. For this purpose, it is effective to add the following $L1$ norm to the objective of CGAN.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{i,gt,n}[||y - G(i, n)||_1] \quad (16)$$

$L1$ norm-based image generation captures the whole image but the blurred details remain a problem. On the other hand, although CGAN-based image generation cannot capture the whole image, it is able to capture the details. By combining these two methods, an image with high accuracy can be generated. Therefore, the objective of Pix2Pix is as follows. w is the weight of $L1$ norm. This parameter can be set during training.

$$G^* = \arg \min_G \max_D \mathcal{L}_{CGAN}(G, D) + w\mathcal{L}_{L1}(G) \quad (17)$$

4.3. Depth Estimation Method

In order to effectively use the optical flow map and RGB image for depth estimation, we need to combine them. The concept of our proposed method is based on the atrous convolution in [33], and we exploit a heat map from the luminance values of the optical flow map and embed it into the RGB image. The heat-map is embedded at a certain number of intervals such that the features in the original RGB image is not lost. Figure 6 shows an example that a optical flow map is embedded into an RGB image. The figure utilizes a sparse optical flow map with a single pixel interval. Each pixel is embedded into the original RGB image.

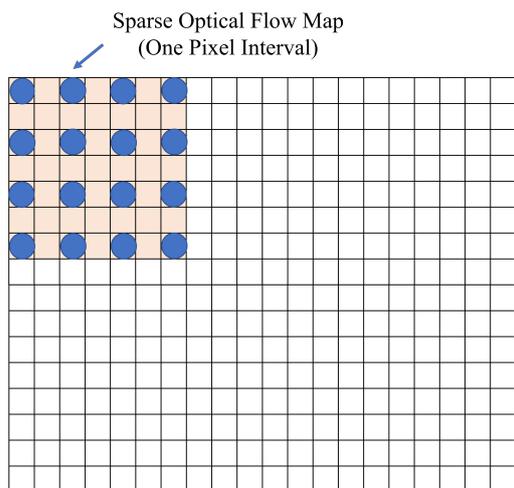


Figure 6. A concept of embedding optical flow map into an RGB image.

The luminance of red, green, and blue towards gray scale luminance is corresponded as shown in Figure 7.

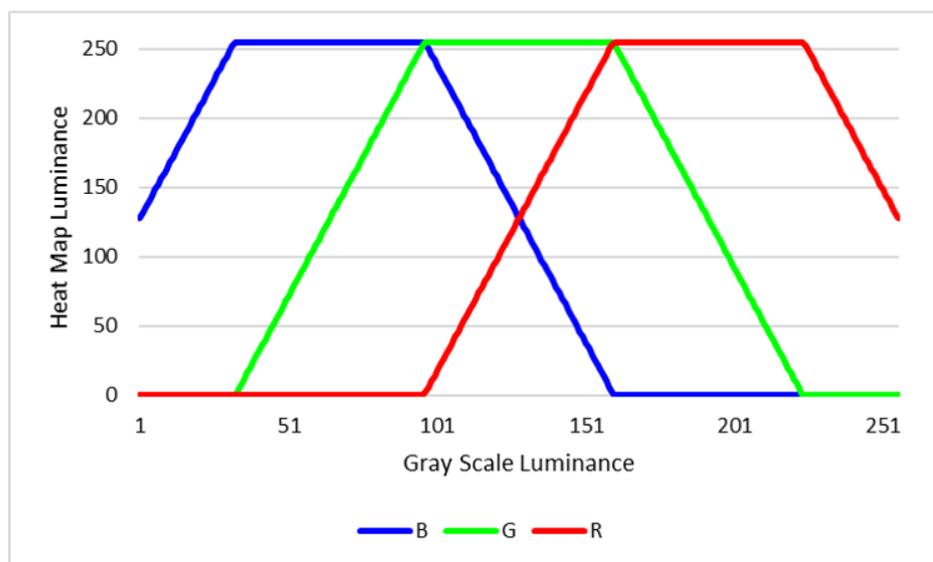


Figure 7. Heat map luminance towards gray scale luminance.

In this work, we embed part of the pixel information of the heat map image into an RGB image to generate a depth image with a single-channel input, shown in Figure 8. This embedding method is expressed in Equation (18).

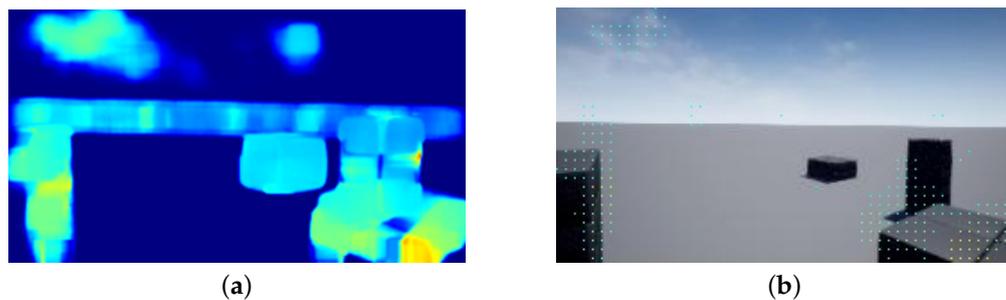


Figure 8. An example of embedding optical flow map into an RGB image: (a) Heat-map of optical flow map, (b) RGB image embedded with a sparse optical flow map.

$$E_{(n,m)} = \begin{cases} M_{(n,m)} & (n \bmod i \neq 0 \cup m \bmod i \neq 0 \cup O_{(n,m)} = 0) \\ O_{(n,m)} & (n \bmod i = 0 \cap m \bmod i = 0 \cap O_{(n,m)} \neq 0) \end{cases} \quad (18)$$

$E_{(n,m)}$ represents the pixel value of the optical flow map at the pixel position of (n, m) embedded in the monocular image. $M_{(n,m)}$ is the pixel value of the monocular image at the (n, m) pixel position and $O_{(n,m)}$ is the displacement of frames in the optical flow map at the (n, m) pixel position. When the $O_{(n,m)}$ is 0, the color of optical flow heat map is deep blue as shown in Figure 8a. We do not use all optical flow pixels to estimate depth since these pixels can be also noises to prevent accurate depth estimation. Therefore we need to select optical flow pixels to use optical flow information efficiently. i is interval between the monocular image the pixel value and optical flow map value. In this way, the optical flow map can be used effectively.

5. Experiments

In this section, we evaluate our method in terms of accuracy, latency and the performance to avoid collisions.

We use Intel Core i7-9700K (32 GB of main memory) and NVIDIA GeForce RTX 2070 SUPER, which is represented in Table 1. Dataset, which are used for training, validation, and testing, have been collected from four maps provided in the AirSim environment; Blocks, City, Coastline, and Neighborhood, where the overviews of the maps are shown in Figure 9.

Table 1. Experimental Environment.

OS	Windows 10 pro
RAM	32 GB 2666 MHz
CPU	Intel Core i7-9700K 3.60 GHz
GPU	NVIDIA GeForce RTX 2070 SUPER 8 GB



Figure 9. Appearance of the maps for training: (a) City environment, (b) Coastline, (c) Neighborhood, (d) Soccer field.

We train our model in the following conditions: the number of epochs is set to 100. The batch size is set to 1, and the lambda of $L1$ norm is set to 100. In the experiments, we have prepared 16,000 pairs of monocular and depth images for each of the maps. 8000 pairs out of 16,000 are used to training our Pix2Pix-based model. The rest of the pairs in monocular and depth images is employed to test out model. In the labelling process, the depth and monocular images are taken through multiple flights with a variety of routes in AirSim beforehand. Figure 10 shows the examples of the inputs and outputs of the model trained with the parameters. Figure 10a shows the RGB images taken by a monocular camera during flights in the four maps of AirSim. At the same time, we obtain the optical flow maps as shown in Figure 10b. From the images, we derive RGB images with embedding an optical flow map in Figure 10c. Compared to the ground truth images in Figure 10d, our proposed method generates depth images as shown in Figure 10e.

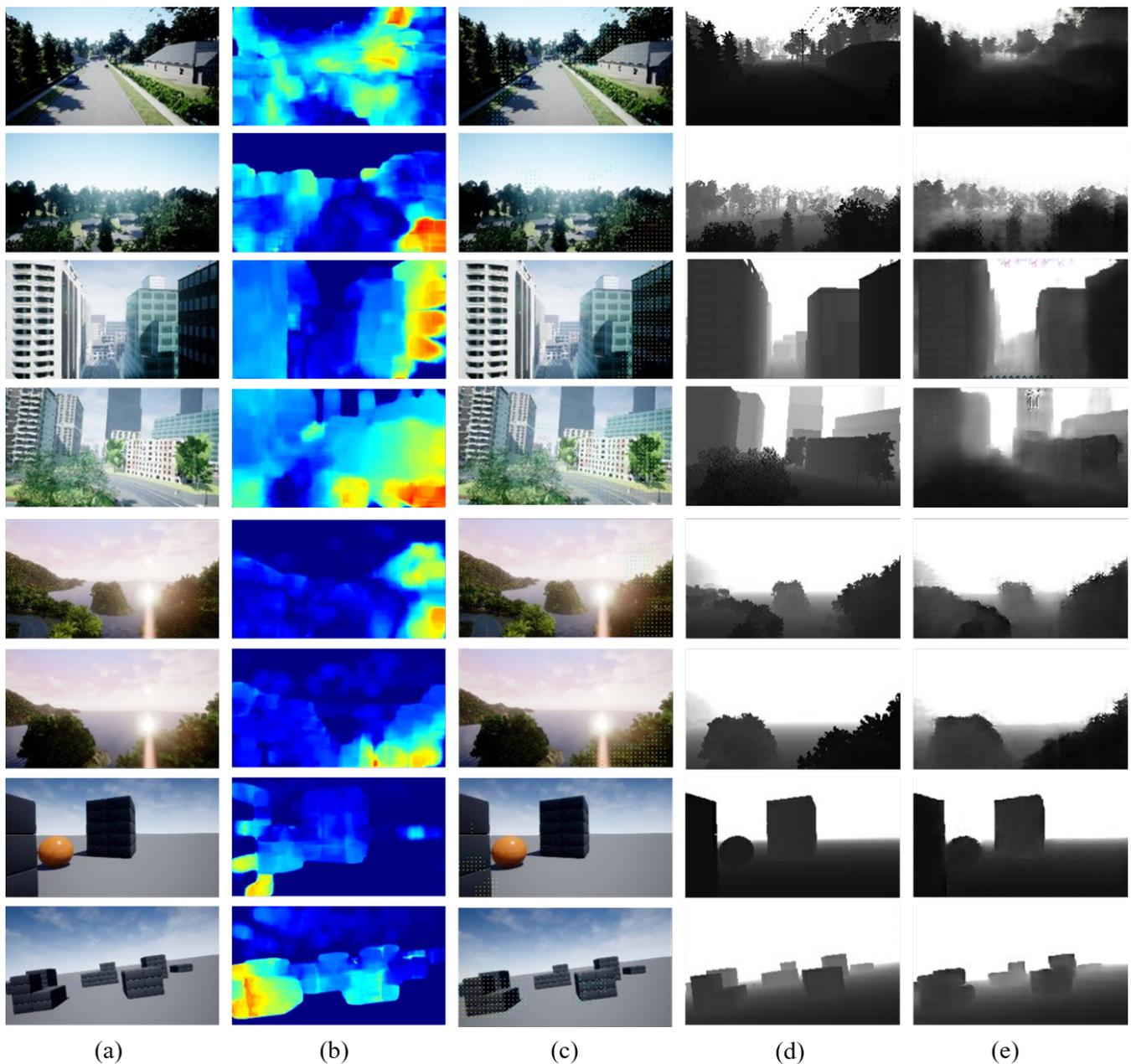


Figure 10. Inputs and outputs: (a) RGB images, (b) Optical flow maps, (c) RGB images embedded with optical flow map, (d) Ground truth, (e) Depth estimation maps.

5.1. Preliminary Evaluation with Different Pixels Interval of Optical Flow Maps

In this experiment, we use six models to investigate the effect of the optical flow maps, and the accuracy and error are compared. One out of six models employs only optical flow maps as input for depth estimation. The others embed the optical flow map into the monocular image at different intervals. The embedding intervals are one, three, five, seven, and nine pixels intervals. We intuitively suppose that the dense pixels of the optical flow map provide much information and achieve higher accuracy than the sparse pixels.

In order to quantify estimation error of models, we use rooted mean squared error (*RMSE*) and absolute relative error (*Rel.*) metrics. Hereby, *RMSE* is obtained by the following equation.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i^{gt} - y_i)^2} \quad (19)$$

y_i^{gt} is ground truth value. y_i is estimation value. N is number of data. $Rel.$ is obtained by the following equation.

$$Rel. = \frac{1}{N} \sum_{i=1}^N \frac{\|y_i^{gt} - y_i\|}{y_i^{gt}} \quad (20)$$

Specifically, the accuracy metrics are defined as:

$$\delta_n = \frac{\text{Card}\left(\left\{y_i : \max\left(\frac{y_i}{y_i^{gt}}, \frac{y_i^{gt}}{y_i}\right) < 1.25^n\right\}\right)}{\text{Card}(\{y_i\})} \quad (n = 1, 2, 3) \quad (21)$$

Table 2 shows the error and accuracy of each model.

Table 2. Preliminary evaluation of our proposed method regarding the error and accuracy.

Pixel Interval	Error (Lower Is Better)		Accuracy (Higher Is Better)		
	<i>RMSE</i>	<i>Rel.</i>	δ_1	δ_2	δ_3
Optical flow only	7.6786	0.3886	0.7416	0.8604	0.9129
1 pixel	6.6258	0.1675	0.8634	0.9376	0.9621
3 pixels	6.0947	0.1397	0.8878	0.9554	0.9758
5 pixels	6.0050	0.1230	0.8923	0.9608	0.9797
7 pixels	6.5064	0.1240	0.8910	0.9605	0.9795
9 pixels	6.7068	0.1335	0.8947	0.9573	0.9762

The results show that the model with the five pixel interval remarks the lowest error and the highest accuracy. In addition, the model trained with only optical flow maps shows the highest error and lowest accuracy. In terms of *RMSE* and *Rel.*, the model trained with only optical flow maps increases 1.6736 points compared to the model with five pixels intervals. As well as the accuracy, the model with five pixels intervals achieves the highest value for each delta metric. The results imply that many pixels intervals might be about to cause over-fitting and lose information of the original RGB images, resulting in the high *RMSE* and *Rel.*, especially over seven pixels intervals. In contrast to the error metrics, the more pixels intervals achieve the improvement of the accuracy.

5.2. Comparison Accuracy between Proposed Method and Related Work

We evaluate our model in terms of the error and accuracy, compared to the model presented in [12]. The compared model is trained without using the optical flow. In other words, this model uses only RGB images to generate depth estimation maps. For our proposed model, we utilize the optical flow embedded into RGB images. The model is selected with five pixels intervals, which represents the lowest *RMSE* and *Rel.* in the Table 2.

Table 3 shows the results of the error and accuracy comparison. Compared to the model without optical flow, we have demonstrated that embedding the optical flow enables to achieve the slightly lower error and higher accuracy.

Table 3. Comparison to the state-of-the-art method using AirSim dataset.

Method	Error (Lower Is Better)		Accuracy (Higher Is Better)		
	RMSE	Rel.	δ_1	δ_2	δ_3
Shimada [12]	5.942	0.1338	0.8871	0.9562	0.9772
Proposed method	6.005	0.1230	0.8923	0.9608	0.9796

As shown in Table 3, in Shimada, T. et al. method, *RMSE* is 5.942, *Rel.* is 0.1338, δ_1 is 0.8871, δ_2 is 0.9562, δ_3 is 0.9772. In proposed method, *RMSE* is 6.005, *Rel.* is 0.1230, δ_1 is 0.8923, δ_2 is 0.9608, δ_3 is 0.9796. in Shimada, T. et al. method, *RMSE* is 5.942, *Rel.* is 0.1338, δ_1 is 0.8871, δ_2 is 0.9562, δ_3 is 0.9772. In *RMSE*, Shimada, T. et al. method was better than proposed method. On the other hand, proposed method is superior in other evaluation indicators.

To confirm whether our method is effective in a real environment, we test our model using the KITTI dataset [24], as shown in Table 4. The KITTI dataset contains RGB images and depth images taken in the real world. We compare our AirSim-based model with other models trained on real images proposed by related works. Although the results of our method are slightly lower than those of association studies based on real model training, it is still a good result.

Table 4. Comparison to the other methods using KITTI dataset.

Method	Error (Lower Is Better)		Accuracy (Higher Is Better)		
	RMSE	Rel.	δ_1	δ_2	δ_3
Eigen et al. [34]	7.156	1.515	0.692	0.899	0.967
Liu et al. [35]	6.986	0.217	0.647	0.882	0.961
Kuznetsov et al. [36]	4.621	0.113	0.862	0.960	0.986
Proposed method	7.605	0.154	0.813	0.958	0.985

5.3. Run Time Evaluation

We also evaluate run time. We evaluate for the servers and the embedded device, which represent NVIDIA RTX 2070 SUPER, Intel Core i7 9700K, and Jetson Xavier NX.

Table 5 shows the results of the run time per image. The slowest run time is shown in the Jetson and represents 0.193 s. In other words, approximately five frames per second can be processed in the Jetson. On the other hand, the result in the NVIDIA RTX 2070 SUPER shows 0.031 s per image. The validation of the results for collision avoidance depends on how long our model can estimate the distance in generated depth images.

Table 5. Runtime to generate an image.

Device	Runtime (s)	
	Non Optical Flow	Optical Flow
NVIDIA RTX 2070 SUPER	0.031	0.134
Intel Core i7 9700K	0.181	0.273
Jetson Xavier NX	0.193	0.297

We have concluded that the processing time is sufficient to avoid collisions in real time. NVIDIA Jetson Xavier NX is a small board computer that can be mounted on a UAV. The weight of Jetson Xavier NX is about 180 g. On the other hand, there is an accurate depth sensor Velodyne HDL-64E used in KITTI dataset [24]. The weight of HDL-64E is 12,700 g [37]. The weight of the other depth sensors which can measure 200 m are also near 1000 g. From the above, Jetson is light enough compared to long range depth sensors like used in KITTI dataset [24]. Jetson is lighter than long range depth sensor. In addition,

unlike attaching such a sensor, the replacement from a base-board into Jetson Xavier NX does not increase the weight so much.

5.4. Collision Rate Evaluation in AirSim Environment

Previously, we have evaluated the accuracy and run time of the proposed method. In this section, we conduct the simulation of a drone flight in AirSim to demonstrate that the proposed method can fly avoiding collision with objects. In order to realize the safe flight of an autonomous drone, it is necessary to plan the path by itself, that is, the drone needs to select the direction so that the drone can avoid colliding with objects. In the experiments, we use a state-of-the-art path planning method for flight control, which is developed in [6]. The work in [6] introduced the method that divides a depth map into multiple sections. The presented method in [6] divides a depth image into 289 overlapped sections (17 rows and 17 columns) as shown in Figure 11.

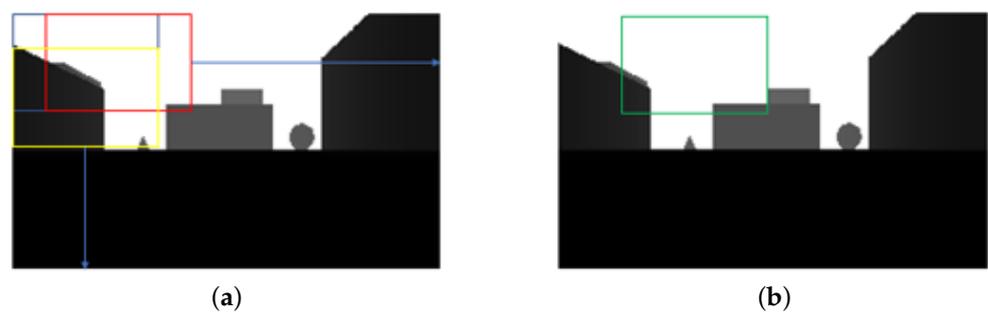


Figure 11. Direction decision from divided sections in [6]. (a) Overlapped section, (b) Section selection.

By dividing into overlapped sections, the drone selects the best section to avoid obstacles and pass safely so that the drone determines the section with the maximum total pixel value. The flight is simulated 400 times in the four maps. The flight scenarios are randomly generated in terms of route, direction, and distance.

We compare the collision rates that the number of collisions account for towards the total number of flights. Hereby, we define the collision rate for a map in the following formula:

$$\text{Collision Rate} = \frac{\text{No. of Collisions}}{\text{No. of Flights (i.e., 400 flights in total)}} \quad (22)$$

Note that we assume that the flight has a collision if the drone collides with an obstacle even once during its flight.

In the experiments, we use the following four methods: The first can measure up to 10 m, which assumes a real depth camera for reasonable price and low weight enough to be equipped on a drone. The second can measure up to 200 m. This method assumes an ideal depth camera, where it can measure by up to 200 m but is too heavy to be mounted on a drone in the real world. This method is used as ground truth depth images for comparison. The third is presented by Shimada, T. et al. [12]. This method inputs a monocular image to generate a depth image through Pix2Pix. The fourth is our proposed method. Our method combines an image with optical flow map into Pix2Pix, and it generates the estimated depth map.

Table 6 shows the results of the collision rate in each map of AirSim. The results show that our proposed method achieves the lower collision rate compared to the method presented in [12]. The depth map for 10 m yields the highest collision rate, and the result explicitly indicates that inaccurate depth images are useless to collision avoidance. The method [12] represents that it achieves the higher collision rate than the proposed method. The results are attributed to depth maps with the low error and high accuracy.

Table 6. Comparison of collision rate.

Map	Collision Rate (%)			
	10 m	200 m	Shimada [12]	Our Method
Blocks	58.75	7.000	17.50	14.50
City environment	73.50	26.00	34.75	34.00
Coastline	70.50	0.250	1.500	1.250
Neighborhood	82.00	7.000	2.500	1.000

6. Discussion

6.1. Evaluation for Effects of Pixels Interval

We discuss that the reason why five pixels interval model achieves the highest accuracy and the lowest error. Figure 12 shows inputs and outputs each model. As shown in Figure 12, the input of the one pixel interval model is filled with optical flow of monocular image features, and the output is distorted. The input of the three pixel interval model is also filled with optical flow pixels. On the other hand, The input of the input of seven pixel interval model and the input of nine pixel model are not enough optical flow pixels. Therefore, five pixels interval model is superior to the others.

Figure 13 shows the input and output of the model which is trained using only optical flow maps. Figure 13a is generated from two adjacent frames of Figure 12a and the previous frame of it. As shown in Figure 13, the output of optical flow model deviates from the ground truth. The reason is that an optical flow map alone cannot accurately capture objects such as buildings if the Pix2Pix-based model is utilized.

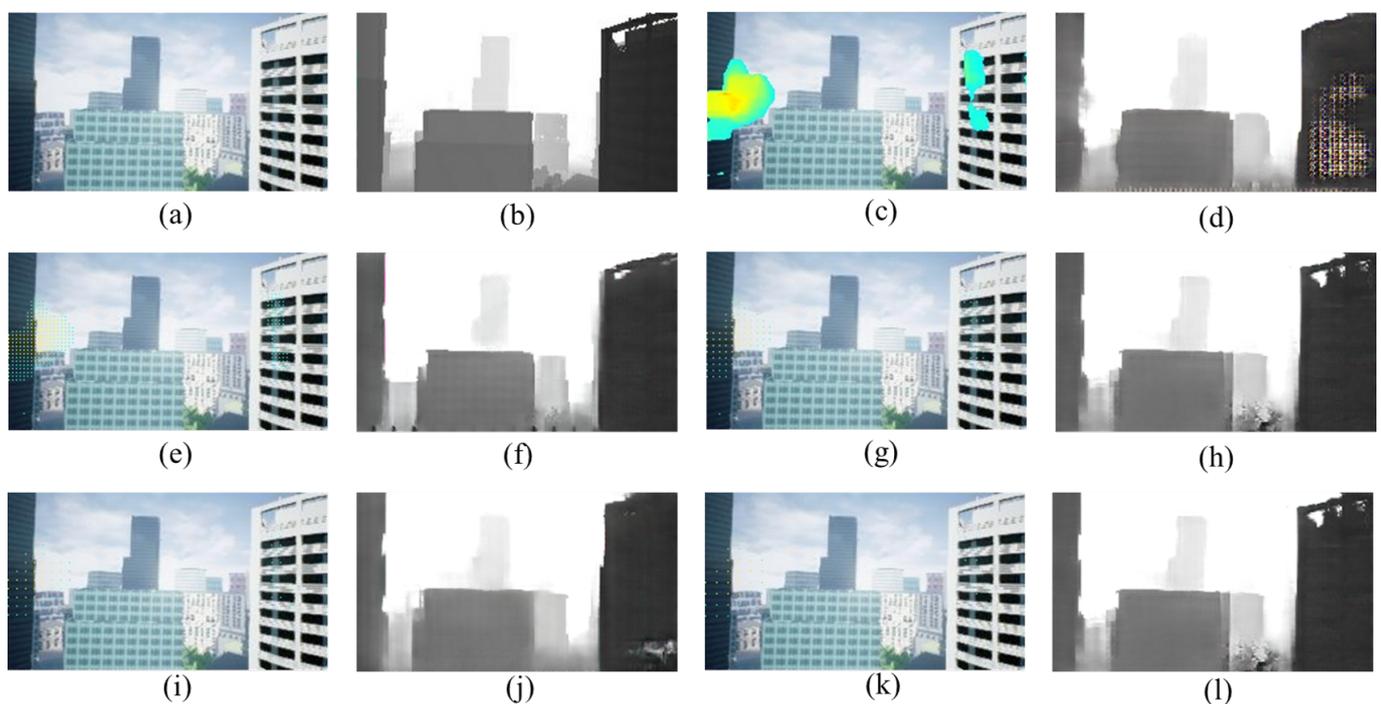


Figure 12. Inputs and outputs of each model: (a) RGB image, (b) Ground truth, (c) One pixels interval model input, (d) One pixels interval model output, (e) Three Pixels interval model input, (f) Three pixels interval model output, (g) Five pixels interval model input, (h) Five pixels interval model output, (i) Seven pixels interval model input, (j) Seven pixels interval model output, (k) Nine pixels interval model input, (l) Nine pixels interval model output.

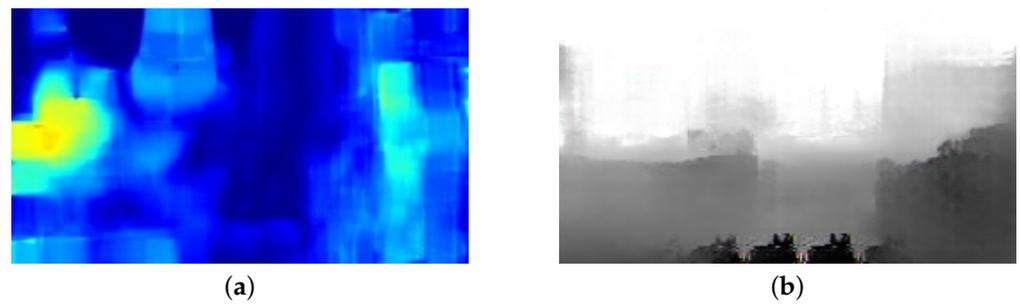


Figure 13. Input and Output of Optical Flow Model: (a) Input, (b) Output.

6.2. Comparison of the Error of Depth Information

Figure 14 shows error distribution.

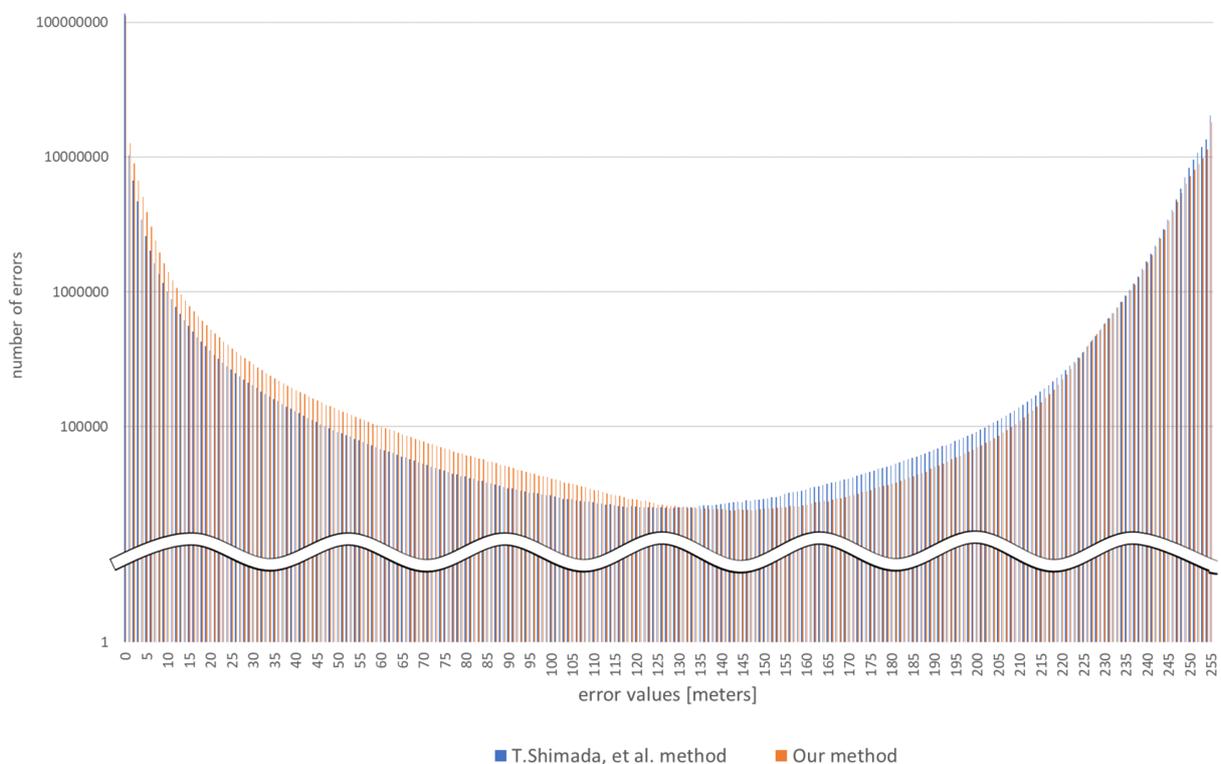


Figure 14. Error Distribution.

In this Figure 14, the horizontal axis shows the value of the error and the vertical axis shows the number of errors. The error value is in meters. The blue bars show the error distribution of Shimada, T. et al. method [12], and the orange bars show the error distribution of the proposed method. As can be seen from this Figure 14, the error of the proposed method is within a smaller range than that of Shimada, T. et al. method [12]. Therefore, it is believed that the proposed method was superior in terms of accuracy and error. In addition, the proposed method has fewer outliers, so the collision rate is considered to be lower than that of Shimada, T. et al. method.

In addition, according to Table 4, although proposed method is higher *RMSE* than other methods, the proposed method is higher accuracy. The reason for this is that the proposed method embeds optical flow pixels in the monocular image, which increases the outliers in those pixels, but improves the accuracy of the surrounding pixels. Therefore, while *RMSE* is degraded due to outliers at that one point, the overall accuracy is high and the value of δ_n is better than other methods. It can be seen from the Table 6 that the outlier at this single point is not a problem for drone collision avoidance.

7. Conclusions

This paper presents the use of Pix2Pix with optical flow to obtain highly accurate depth maps to avoid drone collisions. We have developed an effective way to embed optical flow diagrams in depth estimation. The collision rate of the proposed method is lower than a state of work, over-performing the related works. Even though we used an old image generation method called Pix2Pix, we were able to improve the accuracy of depth estimation by devising a new input image. In addition, we were able to adapt the model trained in the virtual environment to the real world and obtain results comparable to other methods. Even when Pix2Pix with optical flow is used, the results showed that there were few collisions. In order to implement the system on a real drone, it is necessary to install a high-performance computer. Our future work is to study and experiment on how to increase the speed of the system so that it can be used in actual drones. The investigation of generalization performance is also a future task. In addition, we will improve the method more effectively embeds an optical flow map into a monocular image. Finally, we will experiment with real drones and quantitatively evaluate the effectiveness of the proposed method in a real environment.

Author Contributions: Conceptualization, T.S.; Funding acquisition, H.N., X.K. and H.T.; Investigation, T.S.; Methodology, T.S.; Software, T.S.; Supervision, H.N., X.K. and H.T.; Validation, T.S.; Writing—original draft, T.S.; Writing—review and editing, H.N., X.K. and H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partly supported by JSPS KAKENHI Grant Numbers 20K23333 and 20J21208.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors appreciate all the institutions and individuals that have provided support for this paper.

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

References

1. Moffatt, A.; Platt, E.; Mondragon, B.; Kwok, A.; Uryeu, D.; Bhandari, S. Obstacle Detection and Avoidance System for Small UAVs Using A LiDAR. In Proceedings of the IEEE International Conference on Unmanned Aircraft Systems, Athens, Greece, 1–4 September 2020.
2. Hou, Y.; Zhang, Z.; Wang, C.; Cheng, S.; Ye, D. Research on Vehicle Identification Method and Vehicle Speed Measurement Method Based on Multi-rotor UAV Equipped with LiDAR. In Proceedings of the IEEE International Conference on Advanced Electronic Materials, Computers and Software Engineering, Shenzhen, China, 24–26 April 2020.
3. Borenstein, J.; Koren, Y. The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. *IEEE Trans. Robot. Autom.* **1991**, *7*, 278–288. [[CrossRef](#)]
4. Ma, C.; Zhou, Y.; Li, Z. A New Simulation Environment Based on AirSim, ROS, and PX4 for Quadcopter Aircrafts. In Proceedings of the International Conference on Control, Automation and Robotics, Singapore, 20–23 April 2020.
5. Ma, D.; Tran, A.; Ketni, N.; Yanagi, R.; Knight, P.; Joglekar, K.; Tudor, N.; Cresta, B.; Bhandari, S. Flight Test Validation of Collision Avoidance System for a Multicopter using Stereoscopic Vision. In Proceedings of the International Conference on Unmanned Aircraft Systems, Atlanta, GA, USA, 11–14 June 2019.
6. Perez, E.; Winger, A.; Tran, A.; Garcia-Paredes, C.; Run, N.; Ketni, N.; Bhandari, S.; Raheja, A. Autonomous Collision Avoidance System for a Multicopter using Stereoscopic Vision. In Proceedings of the IEEE International Conference on Unmanned Aircraft Systems, Dallas, TX, USA, 12–15 June 2018.
7. Tsuichihara, S.; Akita, S.; Ike, R.; Shigeta, M.; Takemura, H.; Natori, T.; Aikawa, N.; Shindo, K.; Ide, Y.; Tejima, S. Drone and GPS Sensors-Based Grassland Management Using Deep-Learning Image Segmentation. In Proceedings of the International Conference on Robotic Computing, Naples, Italy, 25–27 February 2019.
8. Huang, Z.Y.; Lai, Y.C. Image-Based Sense and Avoid of Small Scale UAV Using Deep Learning Approach. In Proceedings of the International Conference on Unmanned Aircraft Systems, Athens, Greece, 1–4 September 2020.
9. Bipin, K.; Duggal, V.; Madhava Krishna, K. Autonomous Navigation of Generic Monocular Quadcopter in Natural Environment. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015.

10. Lin, Y.H.; Cheng, W.H.; Miao, H.; Ku, T.H.; Hsieh, Y.H. Single Image Depth Estimation from Image Descriptors. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Kyoto, Japan, 25–30 March 2012.
11. Atapour-Abarghouei, A.; Breckon, T.P. Monocular Segment-Wise Depth: Monocular Depth Estimation Based on a Semantic Segmentation Prior. In Proceedings of the IEEE International Conference on Image Processing, Taipei, Taiwan, 22–25 September 2019.
12. Shimada, T.; Nishikawa, H.; Kong, X.; Tomiyama, H. Pix2Pix-Based Depth Estimation from Monocular Images for Dynamic Path Planning of Multirotor on AirSim. In Proceedings of the International Symposium on Advanced Technologies and Applications in the Internet of Things, Kusatsu, Japan, 23–24 August 2021.
13. Fraga-Lamas, P.; Ramos, L.; Mondéjar-Guerra, V.; Fernández-Caramés, T.M. A Review on IoT Deep Learning UAV Systems for Autonomous Obstacle Detection and Collision Avoidance. *Remote Sens.* **2019**, *11*, 2144. [[CrossRef](#)]
14. Valisetty, R.; Haynes, R.; Namburu, R.; Lee, M. Machine Learning for US Army UAVs Sustainment: Assessing Effect of Sensor Frequency and Placement on Damage Information in The Ultrasound Signals. In Proceedings of the IEEE International Conference on Machine Learning and Applications, Orlando, FL, USA, 17–20 December 2018; pp. 165–172.
15. Figetakis, E.; Refaey, A. UAV Path Planning Using on-Board Ultrasound Transducer Arrays and Edge Support. In Proceedings of the IEEE International Conference on Communications Workshops, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
16. McGee, T.G.; Sengupta, R.; Hedrick, K. Obstacle Detection for Small Autonomous Aircraft using Sky Segmentation. In Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 4679–4684.
17. Redding, J.; Amin, J.; Boskovic, J.; Kang, Y.; Hedrick, K.; Howlett, J.; Poll, S. A Real-Time Obstacle Detection and Reactive Path Planning System for Autonomous Small-Scale Helicopters. In Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head, SC, USA, 20–23 August 2007.
18. Trinh, L.A.; Thang, N.D.; Vu, D.H.N.; Hung, T.C. Position Rectification with Depth Camera to Improve Odometry-based Localization. In Proceedings of the International Conference on Communications, Management and Telecommunications (ComManTel), DaNang, Vietnam, 28–30 December 2015; pp. 147–152.
19. Zhang, S.; Li, N.; Qiu, C.; Yu, Z.; Zheng, H.; Zheng, B. Depth Map Prediction from a Single Image with Generative Adversarial Nets. *Multimed. Tools Appl.* **2020**, *79*, 14357–14374. [[CrossRef](#)]
20. Liu, F.; Shen, C.; Lin, G.; Reid, I. Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 2024–2039. [[CrossRef](#)] [[PubMed](#)]
21. Mancini, M.; Costante, G.; Valigi, P.; Ciarfuglia, T.A. J-MOD²: Joint Monocular Obstacle Detection and Depth Estimation. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1490–1497. [[CrossRef](#)]
22. Hatch, K.; Mern, J.; Kochenderfer, M. Obstacle Avoidance Using a Monocular Camera. *arXiv* **2020**, arXiv:2012.01608.
23. Hou, Q.; Jung, C. Occlusion Robust Light Field Depth Estimation Using Segmentation Guided Bilateral Filtering. In Proceedings of the IEEE International Symposium on Multimedia, Taichung, Taiwan, 11–13 December 2017.
24. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision Meets Robotics: The KITTI Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [[CrossRef](#)]
25. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor Segmentation and Support Inference from RGBD Images. In Proceedings of the ECCV 2012, Florence, Italy, 7–13 October 2012.
26. Bhat, S.F.; Alhashim, I.; Wonka, P. Adabins: Depth Estimation Using Adaptive Bins. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 4009–4018.
27. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
28. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
29. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*; Springer: Cham, Switzerland, 2017.
30. Lucas, B.D.; Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver, BC, Canada, 24–28 August 1981.
31. Farnebäck, G. Two-frame Motion Estimation Based on Polynomial Expansion. In Proceedings of the Scandinavian Conference on Image Analysis, Halmstad, Sweden, 29 June–2 July 2003; pp. 363–370.
32. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015.
33. Chen, J.; Zhou, M.; Zhang, D.; Huang, H.; Zhang, F. Quantification of Water Inflow in Rock Tunnel Faces via Convolutional Neural Network Approach. *Autom. Constr.* **2021**, *123*, 103526. [[CrossRef](#)]
34. Eigen, D.; Fergus, R. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale convolutional architecture. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2650–2658.
35. Liu, F.; Shen, C.; Lin, G. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 5162–5170.
36. Kuznetsov, Y.; Stuckler, J.; Leibe, B. Semi-supervised Deep Learning for Monocular Depth Map Prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6647–6655.
37. Lidar, V. *Velodyne Lidar Products*. Available online: <https://velodynelidar.com/products/> (accessed on 1 March 2022).