applied sciences

Article

# General Moving Object Localization from a Single Flying Camera

Kin-Choong Yow [1,*] and Insu Kim [2]

1 Faculty of Engineering and Applied Science, University of Regina, Regina, SK S4S 0A2, Canada
2 Department of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju 61005, Korea; ahinsutime@gmail.com
* Correspondence: Kin-Choong.Yow@uregina.ca; Tel.: +1-306-337-3219

Received: 21 August 2020; Accepted: 30 September 2020; Published: 4 October 2020

**Featured Application: localization of personnel, vehicles, and/or equipment in large open spaces such as farmlands and mines from a single camera mounted on a quadrotor drone.**

**Abstract:** Object localization is an important task in the visual surveillance of scenes, and it has important applications in locating personnel and/or equipment in large open spaces such as a farm or a mine. Traditionally, object localization can be performed using the technique of stereo vision: using two fixed cameras for a moving object, or using a single moving camera for a stationary object. This research addresses the problem of determining the location of a moving object using only a single moving camera, and it does not make use of any prior information on the type of object nor the size of the object. Our technique makes use of a single camera mounted on a quadrotor drone, which flies in a specific pattern relative to the object in order to remove the depth ambiguity associated with their relative motion. In our previous work, we showed that with three images, we can recover the location of an object moving parallel to the direction of motion of the camera. In this research, we find that with four images, we can recover the location of an object moving linearly in an arbitrary direction. We evaluated our algorithm on over 70 image sequences of objects moving in various directions, and the results showed a much smaller depth error rate (less than 8.0% typically) than other state-of-the-art algorithms.

**Keywords:** object localization; flying camera; ego-motion; depth ambiguity; stereo vision; depth recovery

## 1. Introduction

In the visual surveillance of scenes, it is not sufficient to simply detect and track an object [1]. We often need to know the location of the object (or at least the relative location of the object from the camera). This has important applications in finding and locating personnel and/or equipment on a farm [2] or in a mine [3].

There are many methods of estimating the distance of an object from a camera in a scene, such as using reflection of light, laser, radio waves, and ultrasonic waves [4–10]. However, it is more popular to use one or multiple cameras for object localization due to cost considerations [11]. The most general way of object localization from vision is using two fixed cameras to obtain a perspective for a target object. This method is called “stereo vision” [12] and has shown a high accuracy for stationary objects and objects at a near distance [13].

However, the problem with traditional stereo vision algorithms is that, since the distance between two cameras (i.e., the baseline) is short, it is hard to obtain an accurate localization for distant objects as the error rate becomes large. An alternative is to use only one camera that is able to move to a new position to take a second image, and we can make the baseline arbitrarily large [14]. Unfortunately, the object that we are tracking may have moved during that time, rendering traditional stereo vision algorithms useless [15].

In this research, we will address the problem of determining the location of a moving object using a single moving camera. The objectives are two-fold: (1) we want to be able to make the baseline arbitrarily large; (2) we want to remove the depth ambiguity caused by a moving object relative to a moving camera. We approached this problem by mounting a single camera on a quadrotor drone and instructing it to fly in a specific pattern so as to remove the depth ambiguity associated with its relative motion. To maintain generality, we did not assume that we knew the class of the object (e.g., person, vehicle, farming equipment, etc.) or the size, speed, and direction of motion of the object. This allows us to apply this technique to various applications such as locating farm personnel and equipment, mining vehicles, delivery vehicles belonging to logistics supply chains, wild animals in a prairie, etc. In our previous work [16], we showed that, by using three images, we are able to recover the location of an object moving in a direction parallel to the motion of the camera. In this research, we will show that, by using four images, we are able to recover the location of an object moving linearly in an arbitrary direction.

This paper is organized as follows. Section 2 reviews the related works in object localization. The system design of our proposed method is described in Section 3, and the mathematical formulation is discussed in Section 4. Section 5 gives the implementation details and Section 6 shows the evaluation performance of the proposed method. Finally, Section 7 provides some discussion on the localization errors, and Section 8 gives the conclusion and a summary of our proposed approach.

## 2. Related Works

A very popular solution for object localization (and object tracking) is to use drones. The most direct way to achieve this is to get the drone to fly directly to the object and then retrieve its global positioning system (GPS) location. Tracking can be achieved simply by following the object when it moves. However, this very simple approach is inefficient, and it is also not possible to localize multiple objects simultaneously.

A better approach is to use stereo vision on drones. Cigla et al. [17] proposed an onboard stereo system for a drone to perform pursuit as well as sense and avoid. Grewe and Stevenson [18] used a drone-based stereo camera system to assist a low-vision person with environment awareness by detecting a user's heading and direction. Nam and Gun-Woo [19] showed the possibility of using stereo vision with the Inertial Measurement Unit (IMU) to form a tightly coupled Visual-Inertial Navigation System (VINS), which shows a better performance in terms of robustness and accuracy. There is no restriction on the number of cameras for the stereo vision method. Fehrman and McGough [20] performed real-time depth mapping using a low-cost camera array of 16 cameras. Compared to the single camera pair, depth mapping becomes more accurate by increasing the number of camera pairs.

Stereo vision has also been shown to be able to recover distances accurately for near objects. Wan and Zhou [21] introduced a novel stereo rectification method for a dual-fixed Pan-Tilt-Zoom (PTZ) camera system that can recover distance accurately for up to 20 m indoors. Tran et al. [22] proposed a stereo camera system that can perform face detection, with good results for the detection rate (99.92%) in an indoor environment (5 to 20 m). Ramírez-Hernández et al. [23] presented a novel camera calibration method based on the least square method that can improve the accuracy of stereo vision systems for three-dimensional point localization up to a distance of 20 cm. Detecting and localizing moving objects in a stereo vision system is also not a problem. Islam et al. [24] proposed a stereo camera rig with an ultra-wide baseline distance and conventional cameras with fish-eye lenses. They used a unique method to restore the 3D positions by developing a relationship between the pixel dimensions and distances in an image and real-world coordinates. In their research, although the target was moving, the depth extraction results were reasonably accurate. However, their work was conducted in close range (around 2 m) and the object also moved only a little (around 900 mm).

This is the biggest disadvantage of the stereo vision method when localizing objects. Since the algorithm requires the value of the distance between the two cameras (i.e., baseline, which is fixed), it is not good for localizing far objects, as there is only a small disparity between the objects in the two images. We need a much larger baseline to localize far objects. As the baseline gets shorter, the error gets larger. Kim et al. [25] tried to overcome this problem by using multiple drones with a single camera on each of them. They performed research on ground obstacle mapping for Unmanned Ground Vehicles (UGV) without the aid of GPS and using two drones with a single camera on each. Their baseline is changeable, since each drone can be positioned to achieve the desired distance. Xu et al. [26] used three or more unmanned aerial vehicles simultaneously observe the same ground target and acquire multiple remote-sensing images to improve the precision of the target's localization. The multi-unmanned aerial vehicle cooperative target localization method can use more observation information, which results in a higher rendezvous accuracy and improved performance.

The major problem with putting stereo cameras on a drone is the power and the cost. Achtelik et al. [27] described a system that uses stereo vision and laser odometry for drone navigation; however, carrying two cameras at once consumes more power and decreases the payload as well as the duration of flight. For lightweight or low-cost drones, stereo cameras cannot be a solution since the drones cannot afford to even carry the additional camera. Santamaria-Navarro et al. [28] proposed a navigation system for low-cost micro aerial vehicles (MAVs), using two Kalman filters in the extended and error-state flavors. Their system used a single-camera system and is based on low-cost sensors. This is the reason why all the drones on the market currently come with only a single camera.

Another solution is to use pre-determined information to localize objects with a single camera. If we know the information about the target object, such as its size, shape, and color, we can immediately calculate the distance to the object using simple ratio calculations. Chen and Wang [29] showed one good example of such a system. They performed efficient calibration for multiple PTZ cameras by using a single object lying on a horizontal plane with known parameters. Their algorithm works by detecting A4 paper, estimating the angle and altitude of one camera, projecting the A4 paper's location on a 3D space, and then calibrating other cameras with this information.

For drones, this method of using prior information is very simple and has no effects on the duration of flight and payload, since it needs no additional equipment. Zhang and Liu [30] proposed a system where a drone estimates relative altitude from a ground object with (or without) movement if the size of the object is known. Sereewattana et al. [31] performed a depth estimation of color markers for the automatic landing control of Unmanned Aerial Vehicles (UAVs) using stereo vision with a single camera. In their work, the color markers are static (i.e., not moving) and are close to the drone (below 3 m). Sereewattana et al. [32] also extended the concept of automatic landing control for a fixed-wing UAV. Like their earlier work, they took images with a single downward looking camera at different time intervals, and they used static color markers on the flat ground. Nguyen et al. [33] also proposed a novel system to safely land a drone in the absence of GPS signals using a unique marker designed as a tracking target during landing procedures.

Meanwhile, object localization can be performed with a single camera with image analysis. This approach uses feature extraction from images to perform navigation and depth extraction. Huang et al. [34] proposed an accurate autonomous indoor navigation system for a quad copter with a monocular vision. They used information from multiple sensors from the drone as well as a vision-based Simultaneous Localization and Mapping (SLAM) system and Extended Kalman Filter (EKF) to achieve robust and accurate 3D position and velocity estimation. However, they require a large amount of time (about 1 min) with respect to their traveling distance (around 8 m), and their research is confined to an indoor environment which covers a small area.

The solution we are looking for is to use stereo vision principles but with a single-camera system instead. This can be achieved by taking multiple images of the object with a camera that is moving in a certain pattern so as to get the required baseline and image disparity. This system is also free from the decrease in payload and duration of flight, since it does not require any additional hardware gadgets. Thus far, we have found few related works that address these requirements. Bian et al. [35] did propose a novel monocular vision–based environmental perception approach and implemented it in an unmanned aerial vehicle system for close-proximity transmission tower inspection. Their proposed framework comprises tower localization and an improved point–line-based simultaneous localization and mapping framework consisting of feature matching, frame tracking, local mapping, loop closure, and nonlinear optimization. Although their goal is close proximity tower inspection, they could localize the tower fairly accurately.

In our previous work [16], we addressed the simple cases for (1) a stationary object, (2) an object moving parallel to the drone's motion, and (3) an object moving perpendicular to the direction of the drone's motion. For the stationary object case (case 1), by moving the single-camera drone horizontally to a new position, it becomes exactly the same as the classical stereo vision problem. We can recover the distance to the object using just two images. For the case of an object moving perpendicular to the direction of the drone's motion (case 3), it is also exactly the same as the classical stereo vision problem. It does not matter where the starting position of the object is, as we only need two images to recover the position of the object at the time the second image is taken. For the case of the object moving parallel to the drone's motion (case 2), by assuming that the object is moving at a constant speed, we showed that we can recover the distance to the object using three images.

## 3. System Design

### *3.1. System Overview*

In this research, we want to propose a new algorithm for drones with a single camera to localize objects moving in an arbitrary direction. We do not assume any prior knowledge of the object class (e.g., person, vehicle, farm equipment, etc.) or its size, speed, and direction of motion. To achieve the maximal increase in baseline, we instruct the drone to fly in a direction perpendicular to the direction of the camera. As the drone flies, it will take a burst of images at small time intervals. As the drone's motion is known, the location of the drone where each image is taken is completely determined. Using this information and the location of the object in each of the images, we will be able to recover the distance of the object from the camera even though the object's motion is unknown. We demonstrate that, by using four images, we can recover the location of an object moving linearly in an arbitrary direction.

Our algorithm has the advantage that it does not require any additional cameras or apparatus, so it is very effective for lightweight or low-cost drones (e.g., Parrot rolling spider [36]) which have a low payload or low lift force. We demonstrate our algorithm on a pedestrian detector, but we will only use the centroid of the bounding box for our computations and will not make any assumptions about the shape or size of the object. By changing the object detector, our algorithm can work for objects of any class. Once the distance of the object from the drone is known, we can map the object's location to any global coordinate system (assuming that we know the drone's position—e.g., through Global Positioning System (GPS) or WiFi positioning). Our system is also robust to occlusions, as our drone will be taking a series of images as it flies, and we can always select only those that have the object in them. We can also arbitrarily choose the length of the baseline that we want in order to meet the accuracy that we need.

Figure 1 shows an overview of our system. Our system consists of a drone mounted with a single camera at position $(x_d,\ y_d)$. Using the algorithm described later in this section, the drone will compute the distance $Z_d$ and angle $\theta_d$ of an object (e.g., pedestrian) relative to itself, and then use this distance to compute the position of the object $(x, y)$ in global coordinates. We assume that the position of the drone is always known (e.g., through GPS or WiFi positioning, or from an Inertial Navigation System (INS), if available).

After calculating the position of the object, the drone will compute a new position for it to fly to (if the object has moved) so that it will continue to have the object in its view. Figure 2 gives a flowchart of the overall system, including takeoff and landing. Our algorithm is the part in the loop, and it can be repeated as often as necessary.

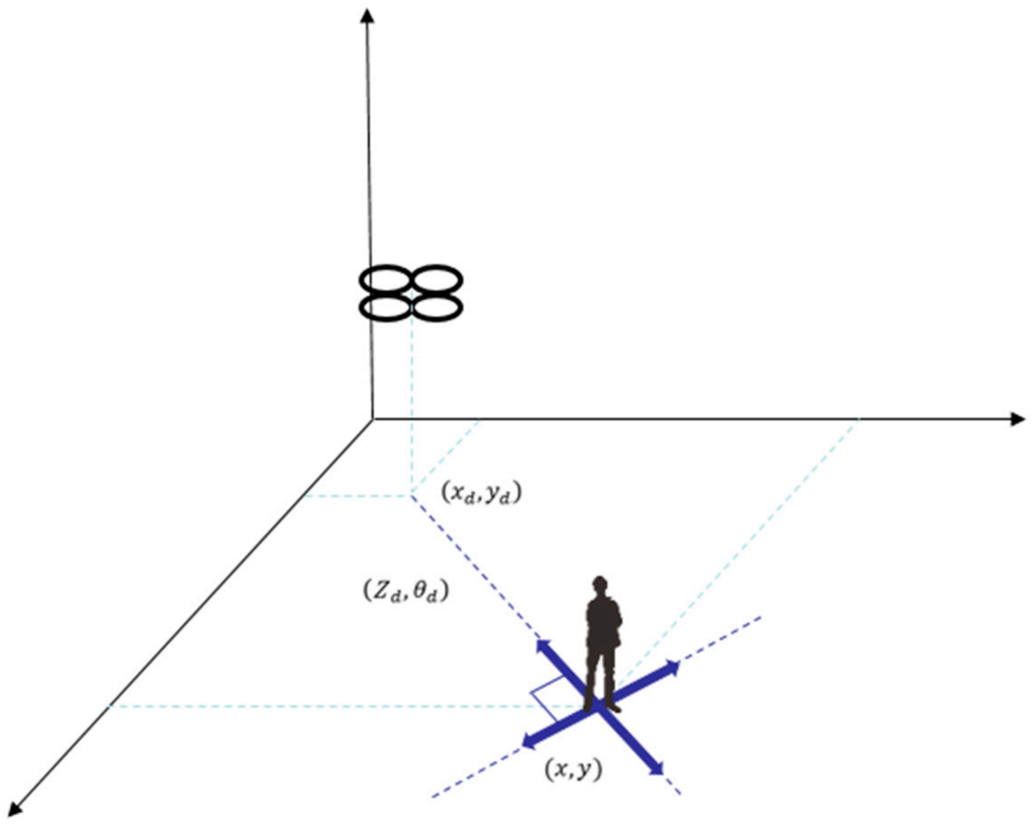

**Figure 1.** Overview of the proposed localization system. The drone is at position $(x_d,\ y_d)$ and the object is at position $(x, y)$. The object is at a distance $Z_d$ and angle $\theta_d$ from the position of the drone.

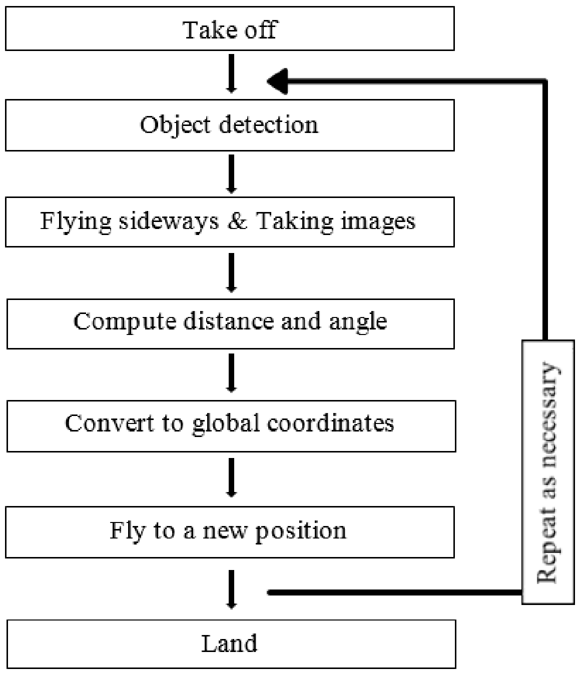
Take off
Object detection
Flying sideways & Taking images
Compute distance and angle
Convert to global coordinates
Fly to a new position
Land
Repeat as necessary

**Figure 2.** Overall system flowchart.

### *3.2. Calibration*

#### 3.2.1. Camera Calibration

In order to obtain an accurate estimation of the depth of the object, the camera needs to be calibrated. We want to be able to recover the angle subtended by an object from the camera axis based on the pixel offset of the object in the image. Figure 3 illustrates the scenario. The drone is at the bottom of the figure with its camera pointing up the page. An object is at the top-right hand corner of the figure, at a depth $Z$ from the camera and a horizontal displacement of $P$ from the camera axis. The object subtends an angle $\theta$ from the optical center axis and causes an image offset of $p$ from the image center. We want to obtain $\theta$ from $p$.

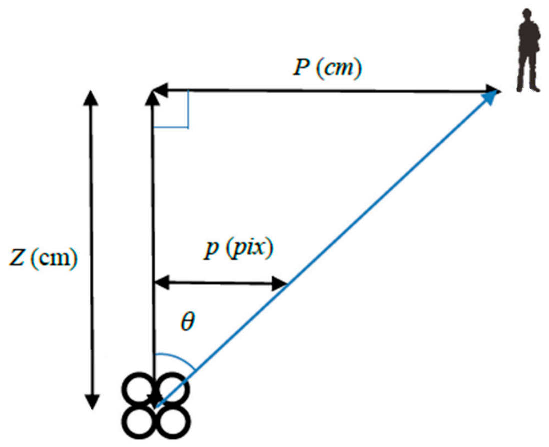

**Figure 3.** Geometry of view angle calibration. The object (pedestrian) is at depth $Z$ and horizontal displacement $P$ from the camera's optical center. It subtends an angle of $\theta$ from the camera axis and has an image offset of $p$ pixels from the image center.

The computation $\theta$ from $p$ is dependent entirely on the focal length of the camera and the image resolution (dots per inch or dpi). To make the computations simpler, we introduce a new parameter $R$ where:

$$\frac{P}{Z} = \frac{p}{R} = \tan\theta. \tag{1}$$

The value of $R$ is a constant for a particular camera. We can obtain this value empirically by placing an object at a depth $Z$ from the camera with a horizontal displacement $P$ from the camera axis in the drone's Field Of View (FOV) and then measuring the image offset $p$. Table 1 shows the data measured from our experiment.

**Table 1.** Calibration of the drone's camera.

| ***Z* (cm)** | ***P* (cm)** | ***p* (pixels)** | ***R* (pixels)** |
|---|---|---|---|
| 92 | 57 | 320 | 516.49 |

With this ratio $R$, when the pixel offset $p$ of an object from the image center is provided, we can immediately calculate the angle $\theta$ subtended:

$$\theta = \arctan\left(\frac{p}{R}\right). \tag{2}$$

#### 3.2.2. Baseline Calibration

Another important set of parameters is the distances between the drone positions at each successive image capture. This represents the baseline between the images. We can determine the drone positions from GPS or WiFi positioning, but these computations are slow and have large errors. It is easier and more efficient to command the drone to perform exactly the same action in each run so that the distances are the same in each run. We can then determine these distances from calibration.

The constant action that the drone is tasked to perform is to take a sequence of images at a constant rate while flying with a preset speed to the left (or right). Any pair of images in this sequence can be used to recover the distance of the object, but to reduce the error in estimating depth the length of the baseline should be as long as possible. This means that we should use the first and the last images in the sequence for the best results. However, as the object may be out of the camera's view as the drone flies, we can use any of the intermediate images in the sequence where the object is last seen.

In our calibration experiments, we allow the drone to takeoff and hover for a few seconds until it has stabilized, and then we send an instruction to the drone to fly to the left (or right) at the maximum speed (i.e., roll angle phi = ±1.0, normalized), while taking a burst of images at a 200 ms interval for a total of 40 images. Figure 4 shows the layout of the baseline calibration. Here, the direction of object movement and drone flight should be equal. For example, if the target object is moving left relative to the drone, then the drone should fly left as well; otherwise, the object will disappear from the image very quickly. These specific values were chosen after numerous tries to give the best tradeoff between the distance flown and the time taken to complete, as the object (pedestrian) may have moved during that time interval. Procedure 1 describes the steps of the baseline calibration procedure.

**Procedure 1**: Baseline calibration

```
Start
1.  Drone takeoff and hover until stabilized.
2.  Instruct the drone to fly to the left (or right) with maximum speed (i.e., roll angle phi $=$ $\pm1.0$,
    normalized).
Repeat
3.  Take an image.
4.  Continue to fly for 200 ms.
Until (number of images taken equals 40)
5.  Match each image taken with the pattern on the wall and measure the distance travelled at the locations
    where each image is taken.
End
```

It is noteworthy to mention that for a quadrotor drone, from a stabilized hovering position it cannot immediately move horizontally. The drone needs to rotate (i.e., "roll") into the direction of intended motion so as to have a component of its lift to generate the force necessary to begin its motion. Due to this, we observe that the position of the drone before the command is sent to the drone (we call this position 0) and the position when the first image is taken (we call this position 1) are exactly the same. The drone was at hovering when the command is sent (i.e., before the drone has even started to rotate (roll)), and the first image was taken just after the drone completed its rotation (roll) to the left or right (but before any horizontal translation takes place—so without any changes in its position). The rest of the images (at positions 2 to $n - 1$) were increasingly distant from the next, due to the inertia and the acceleration of the drone, which needed some time to accelerate to the desired speed.

To measure the length of the baselines, we made the drone fly parallel to a wall with a specific tiling pattern, taking images as mentioned above. From the images, we can determine the positions of the drone where the images were taken and measure the distances between them. We repeat the experiment five times and take the average of these distances as our calibrated baselines. The baseline between any two images in the sequence is simply the sum of the distance values between the two drone positions.

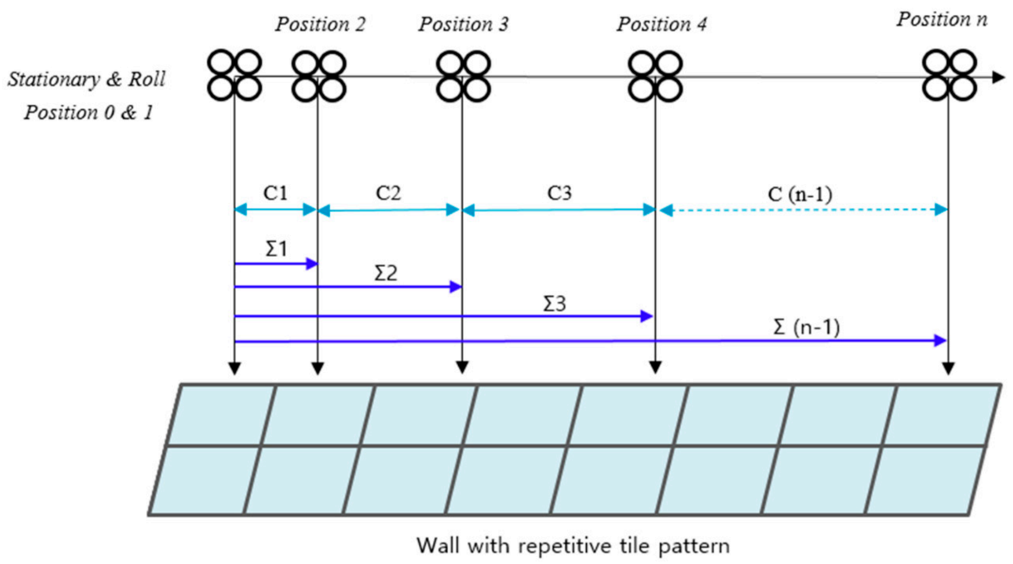

**Figure 4.** Baseline calibration. The drone is flying from the left to the right. The distances C1, C2, etc. are the distances between the positions of the drone where consecutive images are taken. The distances $\sum$ 1 , $\sum$ 2, etc. are the sum of the distances from the drone's starting position to the position where the respective image is taken. $\sum$ 1, $\sum$ 2, etc. correspond to the length of the baseline between the two cameras' positions.

### 3.3. Object Detection

Our system is not restricted to work on only one class of objects. It can be modified to work on any class of objects, but we need to have a reliable object detection algorithm for it. In this paper, we demonstrate our system on pedestrians, and we make use of the Histogram of Oriented Gradients (HOG) pedestrian descriptor from the OpenCV 2.4.10 library ("peopledetect.cpp"). Figure 5 shows an example of the result of the pedestrian detection algorithm in OpenCV, which displays a bounding box over the detected pedestrian.

The pedestrian detector from OpenCV can detect more than one person in a scene. As with any other object tracking algorithm, if we have multiple objects in a scene, we need to make use of additional information (e.g., color, size, motion vectors, etc.) to correctly match the objects in different scenes. False positives can happen too. In our experiments, we will manually select the correct pedestrian if a false positive happens. The design of a more accurate pedestrian detector is not in the scope of our work.

We could have chosen YOLO (You Only Look Once) [37] or used popular object tracking algorithms such as TLD (Tracking-Learning-Detection) [38] or KCF (Kernelized Correlation Filter) [39], etc., in an attempt to make the bounding box more accurate. However, we believe that newer and better detection and tracking algorithms will always be developed, so the accuracy in our method will only increase and not decrease as we use more sophisticated detectors.

### 3.4. Flight Pattern

To obtain the required image disparity, the drone has to fly in a specific pattern. As pointed out in Section 3.2.2, the drone is instructed to perform the action of flying straight to the left or right with maximum speed. However, the drone takes time to reach its maximum speed from its hovering state. Since the drone is instructed to fly with maximum speed, it will be accelerating, which causes the distance travelled between each time interval to increase as time goes by. This is the intended flight pattern in order to make our algorithm work. If the drone flies with constant velocity, and if the object is also moving with a constant velocity, we will get multiple candidate paths due to the geometric similarity between them.

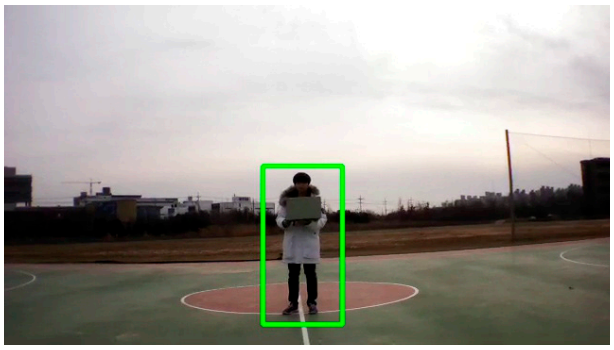

**Figure 5.** Pedestrian detection.

This problem is illustrated in Figure 6. At the bottom is the quadrotor drone moving at a constant speed in a horizontal direction (it does not matter whether it is flying to the left or right). The object we are trying to localize is expressed as a peach-colored rectangle moving in an arbitrary direction (along the peach-colored path). However, if the object is at a further distance and moving with a slower velocity (i.e., the red rectangles along the red path), or if it is at a nearer distance with a faster velocity (i.e., the blue rectangles along the blue path), they will all generate the same image disparity in the image sequence. Of course, the size of the object in the image would be different if it is nearer or further away, but as we do not assume that we know the size of the object, we will not be able to determine which case it is. It is fair to assume that the object is moving at a constant speed; therefore, the drone must be in a state of acceleration so as to avoid this depth ambiguity.

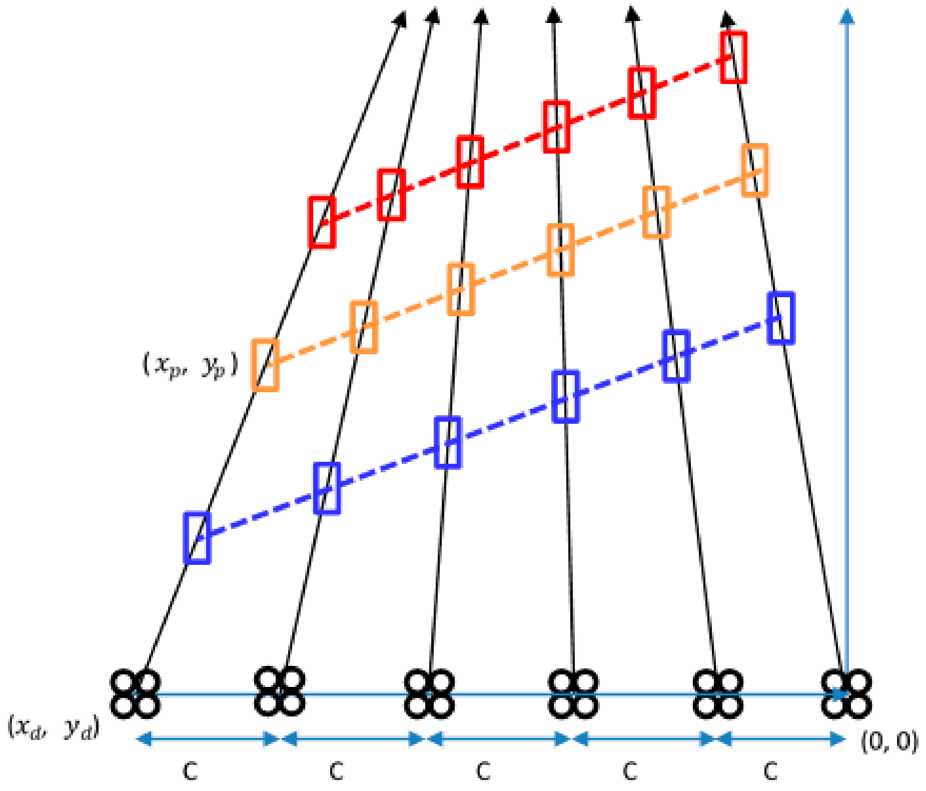

**Figure 6.** Illustration of depth ambiguity. The drone flies from the right to the left at a constant speed, keeping a separation of C between the drone positions $(x_d, y_d)$ where consecutive images were taken. The object (pedestrian) with position $(x_p, y_p)$ is the peach-colored rectangle moving in a straight line with a constant speed. We can see that if the object (pedestrian) is moving along the blue line or red line instead, it will be captured at exactly the same position in all the images.

## 4. Mathematical Formulation

We will now present our formulation for addressing the general moving object localization problem—i.e., estimating the depth of an object moving in an arbitrary direction—from a single flying camera. The simple cases in our previous work [16] are just special cases in the general moving object localization problem, where the motion of the object is limited to certain axes. We will first present the assumptions that will reduce the general moving object localization problem to a linear motion problem, and then we will show that, by using four images, we are able to recover the distance of an object moving linearly in an arbitrary direction. Figure 7 shows the geometry of our localization problem.

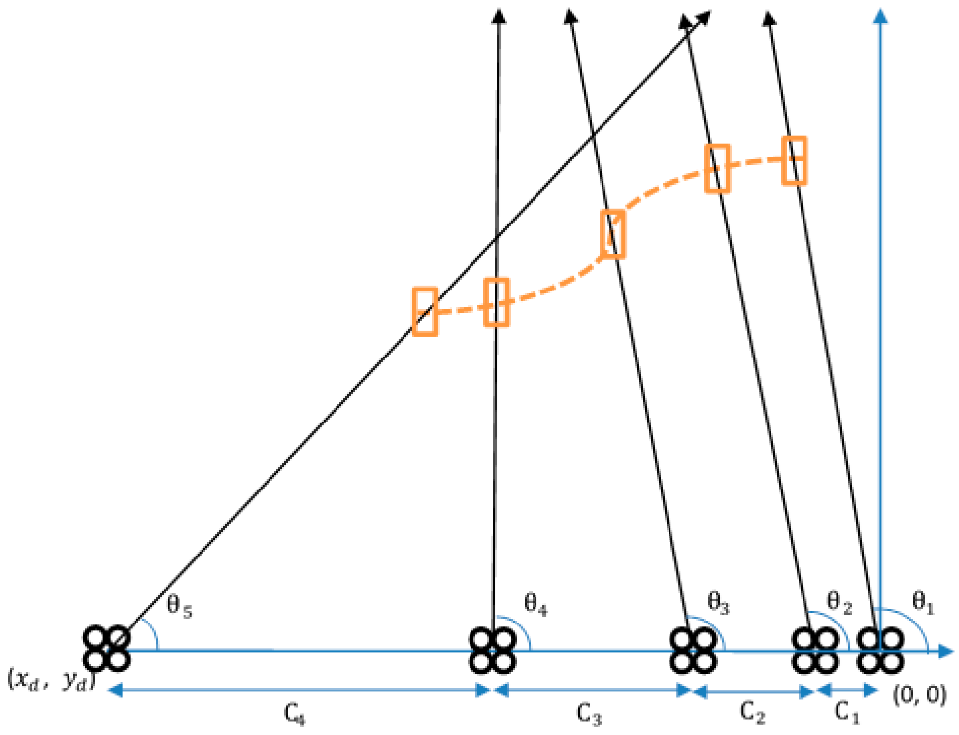

**Figure 7.** Overview of the arbitrary movement of an object. The drone is flying from the right to the left, accelerating towards maximum speed. The distances $C_1$, $C_2$, etc., are increasing as the drone moves with increasing speed. The object may move in an arbitrary fashion, subtending angles $\theta_1$, $\theta_2$, etc., with the camera axes at each position.

The movement of the drone is from the right to the left. Note that the first image (the first black line from the right) is taken when the drone has just completed its rotation, but before any horizontal translation has taken place. Notice also that in subsequent images, the distances between the drones are further and further apart. This is because the drone is accelerating towards its maximum speed.

We make three important assumptions as follows:

1. The drone is moving much faster than the object;
2. The drone's camera's frame rate is high;
3. The drone is always in a state of acceleration.

The first two assumptions will reduce the general motion of the object to that of a linear motion. Usually, we can expect a moving object to follow a straight line of motion and at constant speed, but there are situations where it may not be doing so. If the first two assumptions are true, the movement of the object can be approximated to a linear motion during the time interval that the drone is flying and taking a burst of images of the object. If the object continues to move in a curved or non-linear fashion, the error can be corrected when the drone flies the repeated pattern again, as described in Figure 2.

The third assumption is needed so as to remove the depth ambiguity illustrated in Figure 6. In our experiments, we use a Parrot AR Drone 2.0, which has a maximum speed of 11.11 m/s. We also instructed the drone to capture a sequence of 40 images at 200 ms intervals, which is a high enough frame rate. By the end of the 40 images, the drone would have reached its maximum speed and will not accelerate anymore. Thus, we believe that all these three assumptions are valid and reasonable.

Using all three assumptions, we can represent the object's movement as a straight line and set up the geometry as shown in Figure 8.

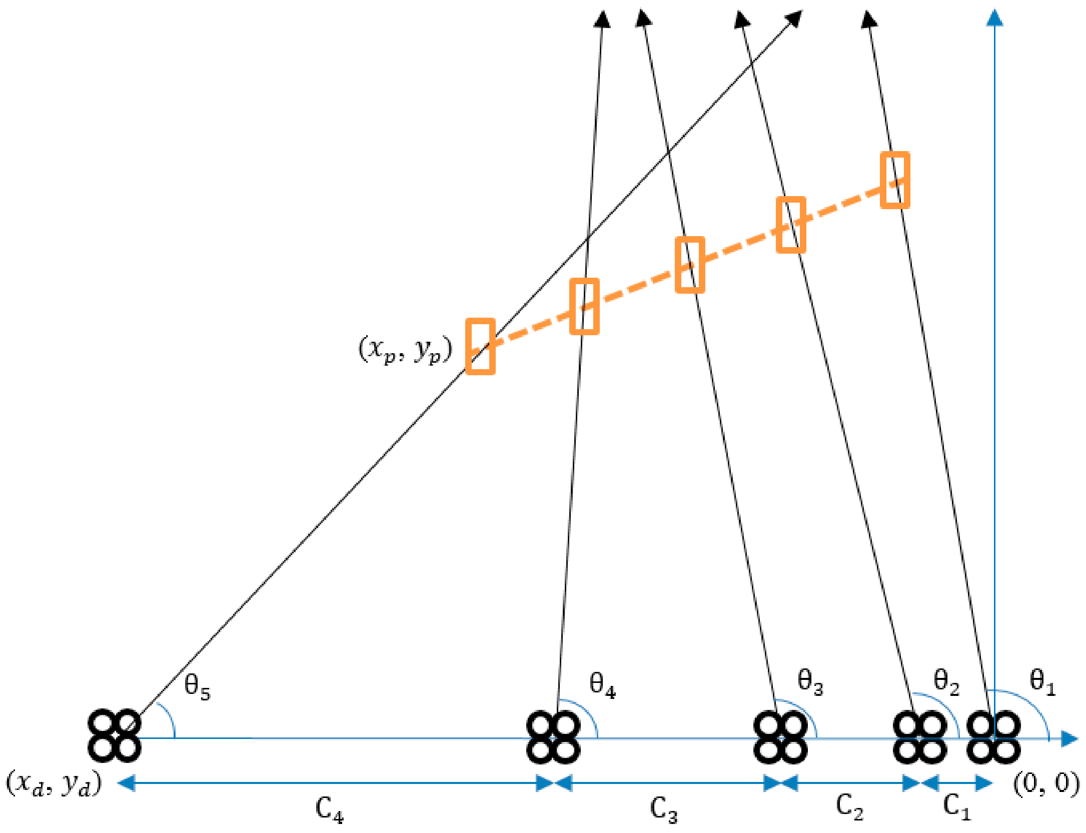

**Figure 8.** Geometry for linear object movement in an arbitrary direction. Here, the object is approximated to move with a constant speed in a straight line.

We can simplify Figure 8 and label our variables as shown in Figure 9. Let the origin (0, 0) be at the bottom right hand corner. Line $L_0$ is the line in the direction of movement of the object, where $(x_1, y_1)$ to $(x_4, y_4)$ are the positions of the object at time periods 1 to 4, respectively. Lines $L_1$ to $L_4$ are the lines joining the center of the object to the image center of the camera for the respective camera positions. $D_A$, $D_B$, and $D_C$ are the distances between the objects in successive time periods in the direction of the object's motion.

To achieve the largest baseline in order to achieve the best accuracy, we should use the first two images and the last two images in the sequence. However, it is possible that the pedestrian may disappear from the drone's view before the end of the sequence. In such situations, we will use the last two images where the pedestrian is still detected (along with the first two images of the sequence) to compute the distance estimation.

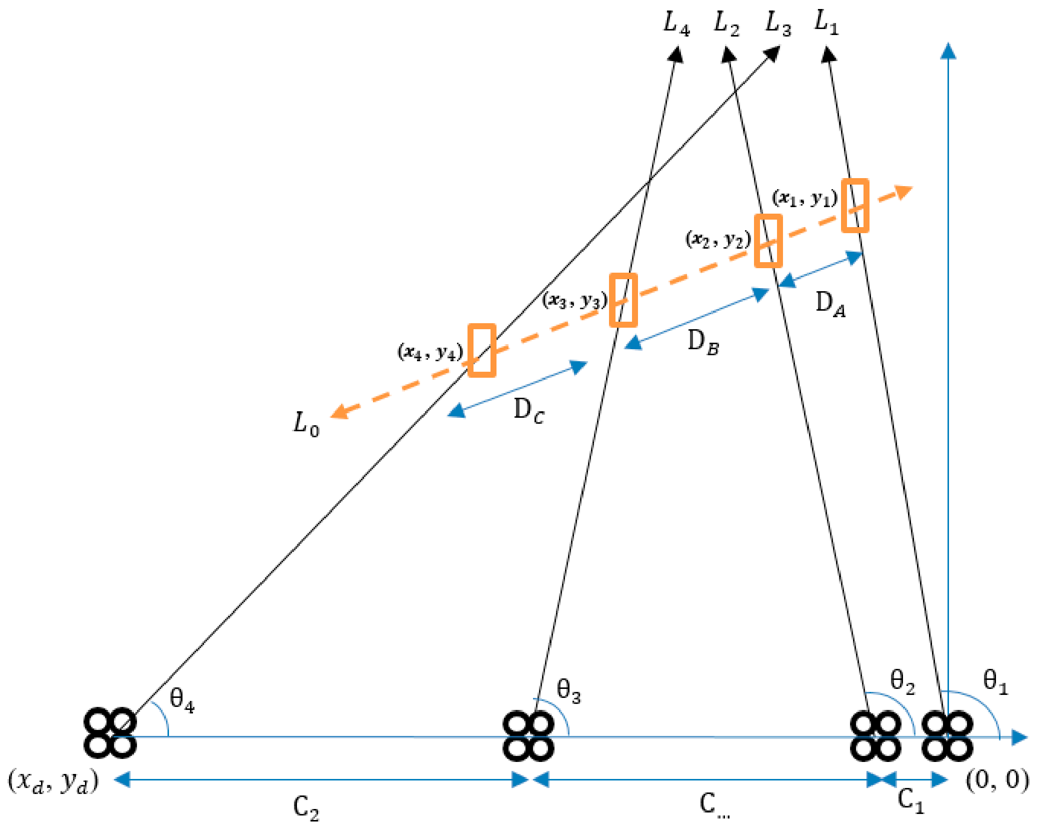

**Figure 9.** Simplified geometry for an object moving linearly in an arbitrary direction. Here, the object is moving along line $L_0$, with distances $D_A$, $D_B$, and $D_C$ between the positions where the respective images were taken. Lines $L_1$ to $L_4$ represent the lines joining the object position, with the camera's optical center in each position where the respective images were taken.

Since our final goal is to obtain $(x_4, y_4)$, we need to determine the equation for line $L_0$ in Figure 9. The equation for line $L_0$ can be expressed as follows:

$$\text{Line } L_0 : y = ax + b. \tag{3}$$

To find this equation, we need to know the equations of the lines $L_1$ to $L_4$ so as to determine the intersection points between them. The equations for lines $L_1$ to $L_4$ can be expressed as follows:

$$\text{Line } L_1 : y = I(x + i), \tag{4}$$

$$\text{Line } L_2 : y = J(x + j), \tag{5}$$

$$\text{Line } L_3 : y = K(x + k), \tag{6}$$

$$\text{Line } L_4 : y = L(x + l). \tag{7}$$

Here, $I$, $J$, $K$, and $L$ are the slopes for each line, and they can be expressed as follows:

$$I = \tan\theta_1,\ J = \tan\theta_2,\ K = \tan\theta_3,\ L = \tan\theta_4\,, \tag{8}$$

and *i*, *j*, *k*, and *l* refer to the *x* axis intercept of the lines, which is essentially the baseline between the respective camera positions and the origin, which can be defined as follows.

$$i = \sum_{1}^{n_1} C_n,\ j = \sum_{1}^{n_2} C_n,\ k = \sum_{1}^{n_3} C_n,\ l = \sum_{1}^{n_4} C_n. \tag{9}$$

where $C_n$ is the distance between successive positions of the drones. Each $n_1$, $n_2$, $n_3$, and $n_4$ refers to the time period in which the image is taken (it is also the image number for that sequence) for the positions "*i*", "*j*", "*k*" and "*l*", respectively. For example, if $n_1$ is 20, the baseline calculation will be:

$$C_1 + C_2 + \cdots + C_{20} = \sum_{1}^{20} C_n = i. \tag{10}$$

Then, the intersections between line $L_0$ and the others can be expressed as follows:

$$(x_1, y_1) = \left(\frac{b - Ii}{I - a}, a\left(\frac{b - Ii}{I - a}\right) + b\right), \tag{11}$$

$$(x_2, y_2) = \left(\frac{b - Jj}{J - a}, a\left(\frac{b - Jj}{J - a}\right) + b\right), \tag{12}$$

$$(x_3, y_3) = \left(\frac{b - Kk}{K - a}, a\left(\frac{b - Kk}{K - a}\right) + b\right), \tag{13}$$

$$(x_4, y_4) = \left(\frac{b - Ll}{L - a}, a\left(\frac{b - Ll}{L - a}\right) + b\right). \tag{14}$$

We can now use the above relations to find an expression for the distances between each successive position of the object. From the geometry in Figure 9, we can obtain the following relations:

$$D_A = \sqrt{\left(\frac{b - Jj}{J - a} - \frac{b - Ii}{I - a}\right)^2 + a^2\left(\frac{b - Jj}{J - a} - \frac{b - Ii}{I - a}\right)^2} = \sqrt{(a^2 + 1)}\left(\frac{b - Jj}{J - a} - \frac{b - Ii}{I - a}\right), \tag{15}$$

$$D_B = \sqrt{\left(\frac{b - Kk}{K - a} - \frac{b - Jj}{J - a}\right)^2 + a^2\left(\frac{b - Kk}{K - a} - \frac{b - Jj}{J - a}\right)^2} = \sqrt{(a^2 + 1)}\left(\frac{b - Kk}{K - a} - \frac{b - Jj}{J - a}\right), \tag{16}$$

$$D_C = \sqrt{\left(\frac{b - Ll}{L - a} - \frac{b - Kk}{K - a}\right)^2 + a^2\left(\frac{b - Ll}{L - a} - \frac{b - Kk}{K - a}\right)^2} = \sqrt{(a^2 + 1)}\left(\frac{b - Ll}{L - a} - \frac{b - Kk}{K - a}\right). \tag{17}$$

Equations (15)–(17) can be reduced further:

$$D_A\left(\frac{b - Kk}{K - a} - \frac{b - Jj}{J - a}\right) = \pm D_B\left(\frac{b - Jj}{J - a} - \frac{b - Ii}{I - a}\right), \tag{18}$$

$$D_B\left(\frac{b - Ll}{L - a} - \frac{b - Kk}{K - a} -\right) = \pm D_C\left(\frac{b - Kk}{K - a} - \frac{b - Jj}{J - a}\right). \tag{19}$$

By simplifying Equations (18) and (19), we can isolate the y-axis intercept "*b*" as an expression using the slope "*a*" from Equation (3). From Equations (18) and (19), we can get $b_1$ and $b_2$, respectively, as follows:

$$b_1 = \frac{T_1 a^2 + T_2 a + T_3}{T_4 a + T_5}, \tag{20}$$

$$b_2 = \frac{t_1 a^2 + t_2 a + t_3}{t_4 a + t_5}, \tag{21}$$

where:

$$T_1 = \frac{D_A}{\pm D_B}(Ii - Jj) - (Jj - Kk),$$
$$T_2 = I(Jj - Kk) - JK(k - j) - \frac{D_A}{\pm D_B}K(Ii - Jj) + \frac{D_A}{\pm D_B}IJ(j - i),$$
$$T_3 = IJK(k - j) - \frac{D_A}{\pm D_B}IJK(j - i),$$
$$T_4 = (K - J) - \frac{D_A}{\pm D_B}(J - I),$$
$$T_5 = \frac{D_A}{\pm D_B}K(J - I) - I(K - J),$$
$$t_1 = (Jj - Kk) - \frac{D_B}{\pm D_C}(Kk - Ll),$$
$$t_2 = \frac{D_B}{\pm D_C}J(Kk - Ll) - \frac{D_B}{\pm D_C}KL(l - k) - L(Jj - Kk) + JK(k - j),$$
$$t_3 = \frac{D_B}{\pm D_C}JKL(l - k) - JKL(k - j),$$
$$t_4 = \frac{D_B}{\pm D_C}(L - K) - (K - J),$$
$$t_5 = L(K - J) - \frac{D_B}{\pm D_C}J(L - K).$$

Since $b_1$ and $b_2$ must be the same value, we can equate (20) and (21) and derive the equation for the slope "$a$". Note that in our linear assumption, the object is moving at a constant speed in a straight line, so $D_A = D_B = D_C$ if four consecutive images in the sequence were used. If, instead, the first two images (i.e., frames 1 and 2) and the last two images where the pedestrian is still detected (say, at frames 18 and 19) were used, we would have $16D_A = D_B = 16D_C$.

The final equation for slope "$a$" is a cubic equation as follows:

$$a^3(T_1t_4 - t_1T_4) + a^2(T_1t_5 - t_1T_5 + T_2t_4 - t_2T_4) + a(T_2t_5 - t_2T_5 + T_3t_4 - t_3T_4) + (T_3t_5 - t_3T_5) = 0. \quad (22)$$

By solving Equation (22), we can get the slope "$a$" of line $L_0$, as well as the intercept "$b$" from either Equation (20) or (21). The Cartesian location of the target object $(x_4,\ y_4)$ can then be determined. We solve the cubic Equation in (22) using Newton's method.

Since we are dealing with a cubic equation, there can be up to three real roots for the slope. Only one of the roots is the correct root and will give a valid object position $(x_4,\ y_4)$ that is consistent with the position and the flight path of the camera. Incorrect roots will give invalid positions, such as negative values of $y_4$ (i.e., behind the camera) or some extreme values of $x_4$.

Note, also, that in our approach fluctuations in elevation do not affect accuracy. The accuracy of the algorithm only depends on the $x$-axis movement.

## 5. Implementation

### 5.1. Hardware

The drone used in this study is the Parrot AR.Drone2 GPS edition [40], as shown in Figure 10. It has a forward-looking 720p HD camera and a vertical QVGA camera. The drone is controlled from an Intel i5 laptop running Windows 8.1 with 4GB of RAM.

**Figure 10.** Parrot AR.Drone2.0.

### *5.2. Software*

The software we used to control the drone is the "CV Drone" package, which is available from Github [41]. The image processing routines were from the OpenCV 2.4.10 library, and the entire system was developed in Microsoft Visual Studio 2013.

### *5.3. Test Environment*

We should choose a test environment where the test subjects can move freely in any direction. In our experiments, we choose to detect pedestrians as they are not confined to moving along fixed roads (as opposed to cars). Our algorithm is sufficiently general and it can work on other object classes if we replace our object detector with an appropriate one for the object class of interest.

Ideally, we should allow our test subjects (i.e., pedestrians in this case) to walk around freely in the test environment, in any direction of their choice. However, regular people do not walk around aimlessly in an environment. Thus, instead of conducting a contrived experiment where we pay the same subject to walk around the test environment in fixed directions, we decided to choose a busy path where lots of pedestrians walk along it with varying walking speeds. To obtain the effect of the pedestrians walking in different directions, we make the drone fly in different directions with respect to the path. Figures 11 and 12 illustrate the ideal layout and the equivalent layout of the experiment.

We decided to record data where the camera moves with five different slopes with respect to the path. As the test environment requires a sufficiently large area for the drone to fly, we selected a large open field (100 m wide and 50 m long, approximately) at the Gwangju Institute of Science and Technology with a path that is used frequently by pedestrians to cross from one side of the campus to the other. Figure 13 shows the layout from a Google map of the area. We capture over 70 image sequences with different pedestrians moving at their own natural pace.

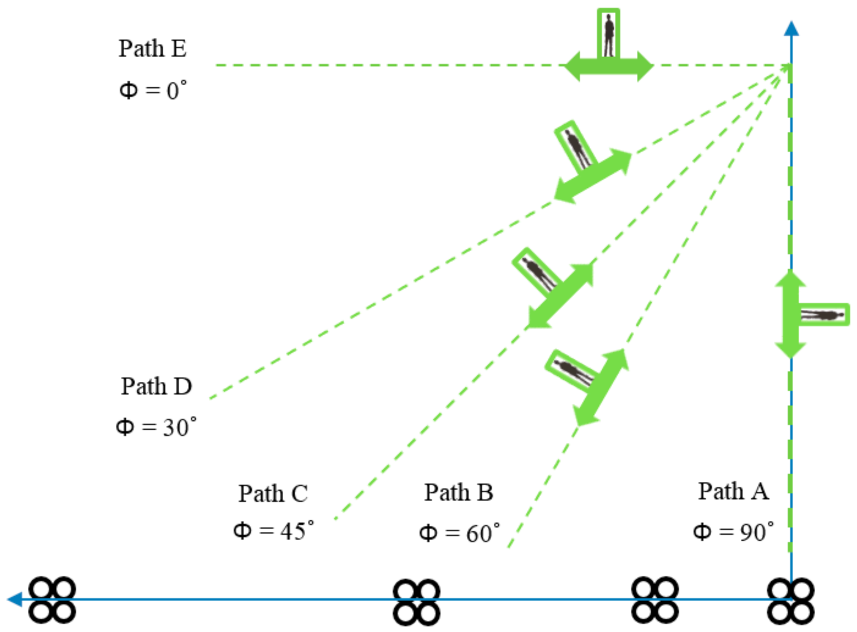

**Figure 11.** Ideal layout for subjects moving in an arbitrary direction.

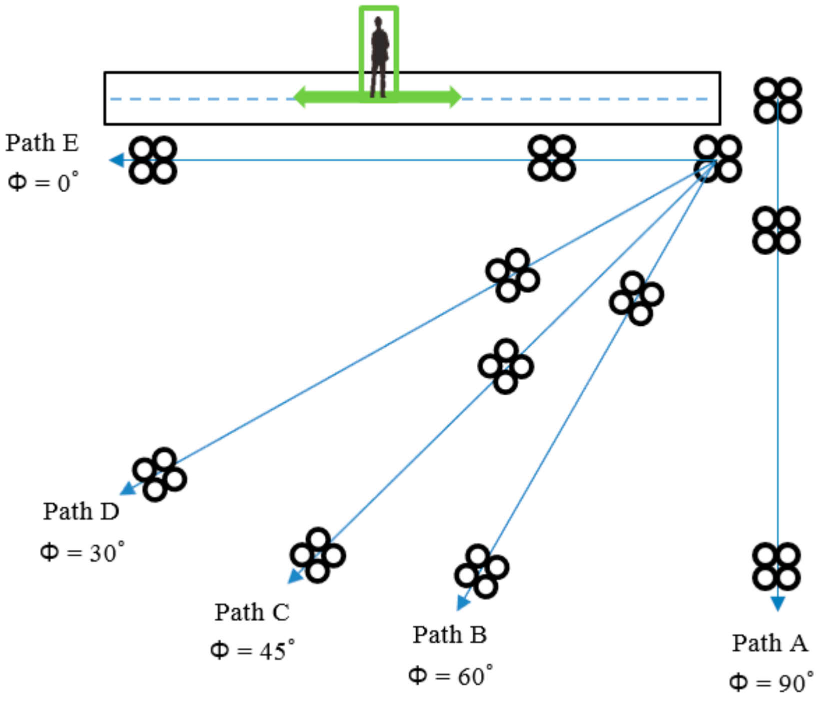

**Figure 12.** Equivalent layout for subjects moving in an arbitrary direction. Here, the subjects always move along a fixed path. The camera will move at different angles with respect to the path.

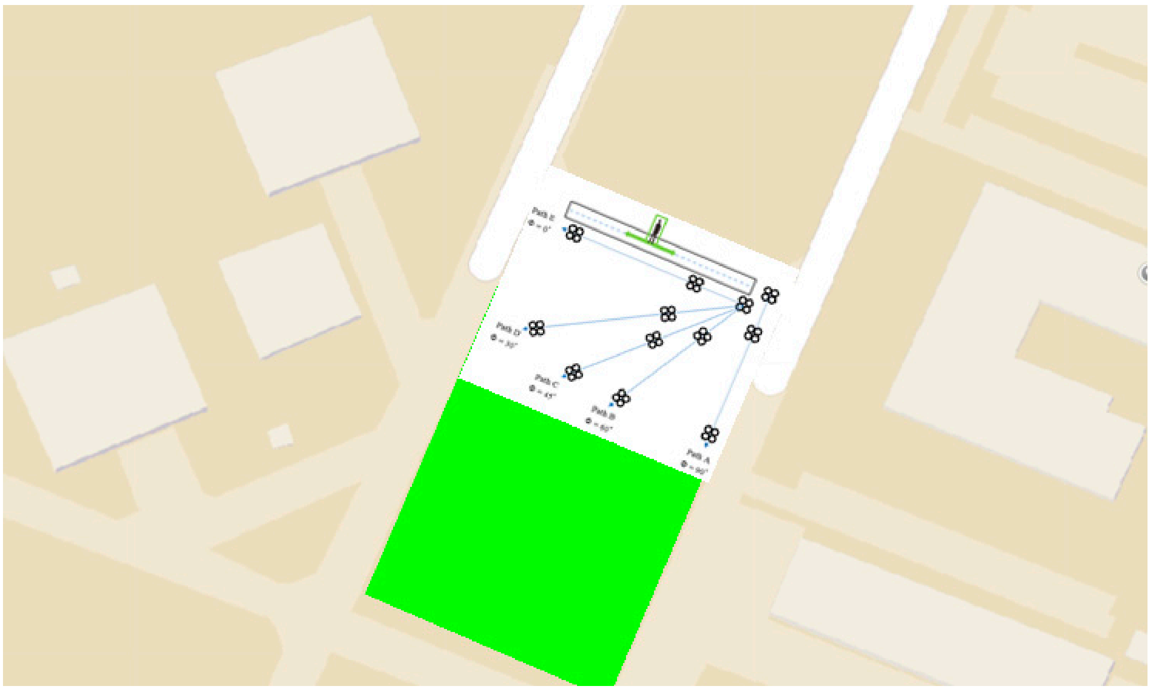

**Figure 13.** Test environment for the pedestrian moving in an arbitrary direction.

## *5.4. Performance Indicators*

For each of the experiments, we compute the depth error:

$$Depth\ error\ (\%) = \frac{|Z - Z\prime|}{Z}, \tag{23}$$

where $Z$ refers to the actual depth measured from the position of the drone at the last image of the four images used in the computation, while $Z\prime$ indicates the computed depth using our algorithm. We will also compute the actual position error using the following equation:

$$Position\ error\ (m) = \sqrt{(x - x\prime)^2 + (y - y\prime)^2}, \tag{24}$$

where $(x, y)$ refers the actual position of a pedestrian, while $(x\prime, y\prime)$ indicates the computed position.

## 6. Results

### 6.1. Performance Evaluation

For each image sequence, we took 40 images. The distance from the drone to the pedestrian varies between 5 and 20 m. The speed of each pedestrian is assumed to be constant, while the exact value is unknown. Various examples of the images that were captured by the drone are shown in Figures 14–18. For each figure, five image frames (frame 0, 1, 10, 18, and 19) were shown. Frame 0 represents the image as seen by the drone before the command is sent to the drone.

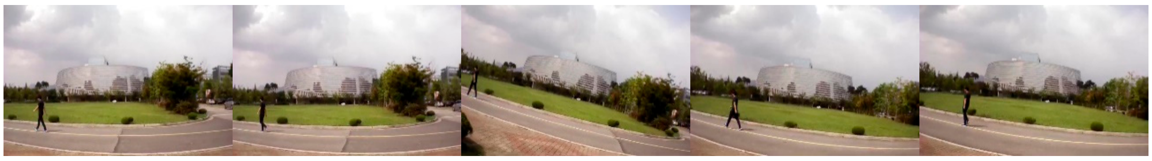
**Figure 14.** Image sequence of a pedestrian moving parallel to the camera's motion.

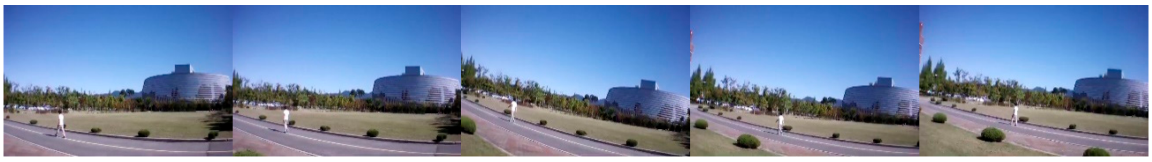
**Figure 15.** Image sequence of a pedestrian moving with an angle of 30 degrees.

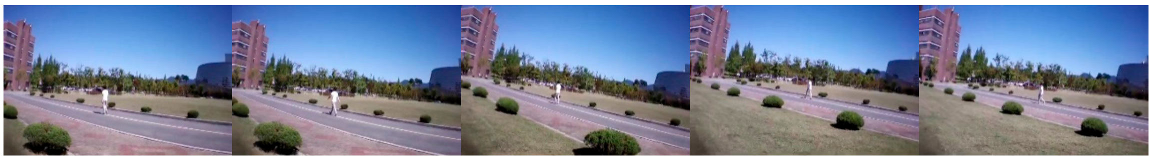
**Figure 16.** Image sequence of a pedestrian moving with an angle of 45 degrees.

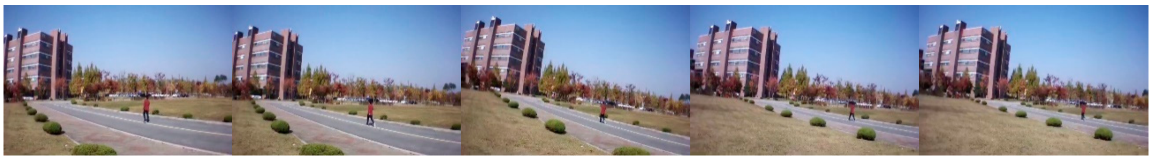
**Figure 17.** Image sequence of a pedestrian moving with an angle of 60 degrees.

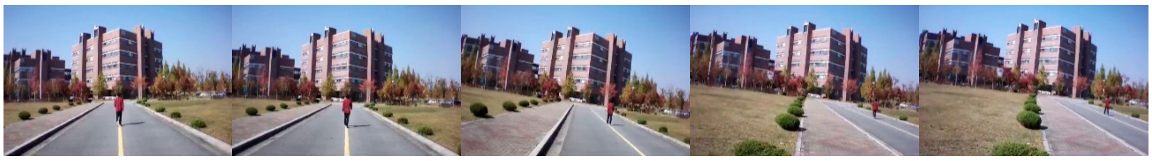
**Figure 18.** Image sequence of a pedestrian moving perpendicular to the camera's motion.

In each sequence, it is possible that the pedestrian will disappear from the drone's view. In such situations, we will use the last two images where the pedestrian is still detected (along with the first two images of the sequence) to compute the distance estimation. In this way, we keep the baseline as large as possible so as to achieve the most accurate results. For each experiment, we computed the depth error using Equation (23), and we used Equation (24) to obtain the position error. The results are shown in Tables 2–6.

**Table 2.** Pedestrian moving parallel to the camera's motion.

| **Φ = 0°** | **Estimated X (cm)** | **Estimated Y (cm)** | **Actual X (cm)** | **Actual Y (cm)** | **Depth Error (%)** | **Position Error (cm)** |
|---|---|---|---|---|---|---|
| Trial 1 | −818.5 | 774.8 | −646.1 | 759.1 | 1.1 | 173.1 |
| Trial 2 | −839.5 | 859.3 | −833.6 | 837.7 | 1.6 | 22.4 |
| Trial 3 | −714.2 | 795.2 | −724.4 | 869.5 | 5.6 | 75.0 |

**Table 3.** Pedestrian moving with an angle of 30 degrees.

| **Φ = 30°** | **Estimated X (cm)** | **Estimated Y (cm)** | **Actual X (cm)** | **Actual Y (cm)** | **Depth Error (%)** | **Position Error (cm)** |
|---|---|---|---|---|---|---|
| Trial 1 | −1237.9 | 1532.0 | −1247.3 | 1570.1 | 1.8 | 39.2 |
| Trial 2 | −918.6 | 1459.2 | −926.0 | 1493.6 | 1.9 | 35.2 |
| Trial 3 | −992.0 | 1833.5 | −959.9 | 1681.7 | 7.7 | 155.2 |

**Table 4.** Pedestrian moving with an angle of 45 degrees.

| **Φ = 45°** | **Estimated X (cm)** | **Estimated Y (cm)** | **Actual X (cm)** | **Actual Y (cm)** | **Depth Error (%)** | **Position Error (cm)** |
|---|---|---|---|---|---|---|
| Trial 1 | −965.9 | 1797.2 | −967.9 | 1830.0 | 1.4 | 32.9 |
| Trial 2 | −862.8 | 1799.8 | −810.2 | 1432.7 | 21.3 | 370.8 |
| Trial 3 | −534.0 | 1177.8 | −371.8 | 1767.7 | 28.4 | 611.8 |

**Table 5.** Pedestrian moving with an angle of 60 degrees.

| **Φ = 60°** | **Estimated X (cm)** | **Estimated Y (cm)** | **Actual X (cm)** | **Actual Y (cm)** | **Depth Error (%)** | **Position Error (cm)** |
|---|---|---|---|---|---|---|
| Trial 1 | −491.4 | 2057.3 | −486.9 | 2082.6 | 1.1 | 25.7 |
| Trial 2 | −218.6 | 2121.0 | −225.8 | 2081.2 | 1.9 | 40.5 |
| Trial 3 | −392.2 | 1730.0 | −359.2 | 1852.6 | 6.0 | 127.0 |

**Table 6.** Pedestrian moving perpendicular to the camera's motion.

| **Φ = 90°** | **Estimated X (cm)** | **Estimated Y (cm)** | **Actual X (cm)** | **Actual Y (cm)** | **Depth Error (%)** | **Position Error (cm)** |
|---|---|---|---|---|---|---|
| Trial 1 | 139.6 | 1450.1 | 134.0 | 1439.1 | 0.8 | 12.4 |
| Trial 2 | −221.7 | 1920.6 | −227.5 | 1898.6 | 1.1 | 22.8 |
| Trial 3 | −46.3 | 1768.6 | −76.3 | 1673.9 | 5.6 | 99.4 |

Except for the case of pedestrian moving with an angle of 45 degrees, all the experiments yielded good results with a <8.0% depth error rate. For the case of the pedestrian moving with an angle of 45 degrees, there is one trial which gave a good accuracy with a depth error of 1.1%. In the other two trials, the algorithm had difficulty detecting the pedestrian near the end of the image sequence because the color of the clothing worn by the pedestrian was too similar to the background color. As a result, only the earlier images in the sequence were used, which led to a small baseline and thus a high error.

### *6.2. Comparison with Other Techniques*

We compare our algorithm with the classical stereo vision technique of computing image disparity and then calculating the distance from the image disparity. This comparison is performed only for the stationary pedestrian case. We used two images of the pedestrian taken from the drone's camera placed 1 m apart (not in flight) and we used the same pedestrian detector (based on the HOG descriptor) to find the boundary boxes and then the centroids. The image disparity between the two centroids is then used to compute the distance in the same way as classical stereo vision. Table 7 shows the results.

**Table 7.** Comparison with the classical method (stationary pedestrian).

| | **Classical Method** | | **Our Method** | |
|---|---|---|---|---|
| **Actual Depth, Z (m)** | **Measured Depth, Z′ (m)** | **Depth Error Rate (%)** | **Measured Depth, Z′ (m)** | **Depth Error Rate (%)** |
| 2.75 | 2.45 | 11.0 | 2.35 | 14.5 |
| 5.50 | 7.18 | 31.4 | 6.01 | 9.3 |
| 7.25 | 8.79 | 21.2 | 7.51 | 3.6 |
| 9.00 | 10.01 | 11.2 | 8.96 | 0.4 |

While our method performs worse at a short depth (2.75 m), it actually works better than the classical method for the longer depths. This is because at shorter depths, the object disappears from the drone's view rather quickly, so we can only use the earlier images in the sequence, which correspond to a small baseline. For longer depths, the object stays in view of the drone's camera as the drone flies over a longer distance, thus generating a larger baseline.

We note that at the shorter distances of 5.5 and 7.25 m, the classical stereo vision method has a large depth error rate. The reason for this is because the image feature used in the computation is only the centroid of the bounding box from the pedestrian detector. There are errors in computing this centroid, and when this is divided by a small denominator (i.e., the shorter distances), it results in a larger relative error.

We further compare our method with the method proposed by Sereewattana et al. [31]. In their approach, they placed colored markers on their target and used a drone with a single camera to fly to a second location to obtain the image disparity of the colored marker, so as to estimate the depth for UAV landing control. Their experiments were conducted only in the range of 2 m. In this evaluation, we extend their method by placing a larger marker on our object so that it can be seen from further away, and we made the drone fly sideways for a distance of 15 m to capture the second image. In our method, the distance travelled by our drone after taking 40 images at 200 ms is 15.5 m, which makes the comparison fair. However, in our method we use the HOG pedestrian detector to detect the bounding box and the centroid, while in Sereewattana et al.'s method they use color thresholding to detect the centroid of their circular marker. Table 8 shows the results.

**Table 8.** Comparison with other systems (stationary case).

| | **Sereewattana et al. [31]** | | **Our Method** | |
|---|---|---|---|---|
| **Actual Depth, Z (m)** | **Measured Depth, Z′ (m)** | **Depth Error Rate (%)** | **Measured Depth, Z′ (m)** | **Depth Error Rate (%)** |
| 2.75 | 2.41 | 12.4 | 2.35 | 14.5 |
| 5.50 | 4.93 | 10.4 | 6.01 | 9.3 |
| 7.25 | 7.84 | 8.1 | 7.51 | 3.6 |
| 9.00 | 9.35 | 3.9 | 8.96 | 0.4 |

The results showed that our algorithm can estimate depth with an equal or better accuracy than Sereewattana et al.'s method. As both the baselines are almost the same, the depth error rates are similar, with our method being slightly better at longer distances but poorer at shorter distances. The difference is caused by them using a circular marker, which they can extract more accurately at shorter distances but not at longer distances.

## 7. Discussion

Our algorithm has been shown to work well and have a much lower depth error rate (less than 8.0% typically) than related state-of-the-art methods. Meanwhile, there are exceptional cases where the depth accuracy of our algorithm drops. The following are possible causes of errors.

### *7.1. Near Constant Speed Problem*

As mentioned in Section 3.4, depth ambiguity happens when both the pedestrian and the drone travel with constant speed. Thus, the drone must be in a constant state of acceleration until it takes the entire image sequence. However, we observe that at around frame 25 of a sequence (any sequence), the acceleration of the drone has dropped to a fairly small value (i.e., it is almost reaching constant velocity). Figure 19 shows the images of frames 35 to 39 when the drone moves with an almost constant speed.

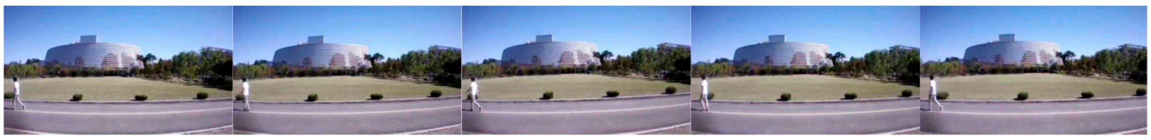

**Figure 19.** Depth ambiguity due to the almost constant speed of the drone.

In Figure 19, the pedestrian appears at the same location in multiple images due to the almost constant speed of the drone. Thus, it is important not to choose baseline fragments where the drone has almost reached a constant speed. This problem can be mitigated by tuning the flight parameters of the drone so that it accelerates at a slower rate, thus reducing the likelihood of attaining an almost constant speed before the end of the sequence. Alternatively, the drone can fly in a sine wave pattern (i.e., it will accelerate to almost constant speed and then begin to decelerate towards zero, and the cycle continues) so that a significant acceleration is always maintained.

### *7.2. Wind Interference*

Wind is the most critical and typical factor which affects the accuracy of the algorithm directly. The presence of wind can change the position of the drone in its flight. A tail wind will move the drone further along its axis, and a head wind will slow it down or even move it backwards. This is significant, because it changes the baseline between the camera positions. Figure 20 shows the situation when a sudden wind affects the drone's position and attitude while it is still hovering.

This situation can be mitigated if the wind speed or direction is known—e.g., from a wireless anemometer. We can find the component of the wind in the direction of the drone's flight and then add or subtract it from the drone's actual speed. This will allow us to compute a multiplier for the distance travelled by the drone so as to compensate for the effect of the wind. Another alternative is to make use of positioning systems such as GPS or WiFi positioning to provide the correction needed due to the wind. However, such positioning systems can be quite slow, so this alternative will only work for very slow-moving or stationary objects.

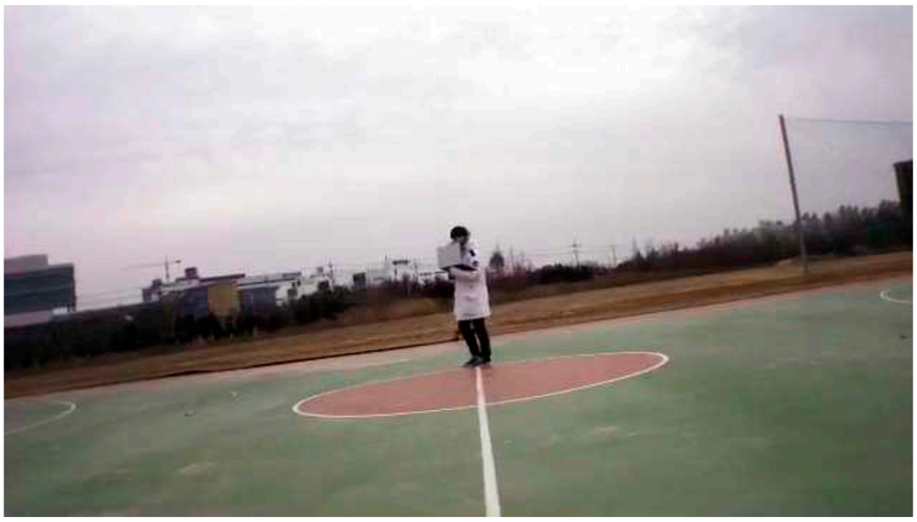

**Figure 20.** Sudden wind interruption.

### *7.3. Low Resolution*

The resolution of the forward camera in AR.Drone is enough for localizing close-range pedestrians. However, for pedestrians at far distances (e.g., >50 m), the resolution is not enough for accurate pedestrian detection. In this case, the center location of the pedestrian could be wrong. Figure 21 shows the error caused by poor resolution when localizing a pedestrian.

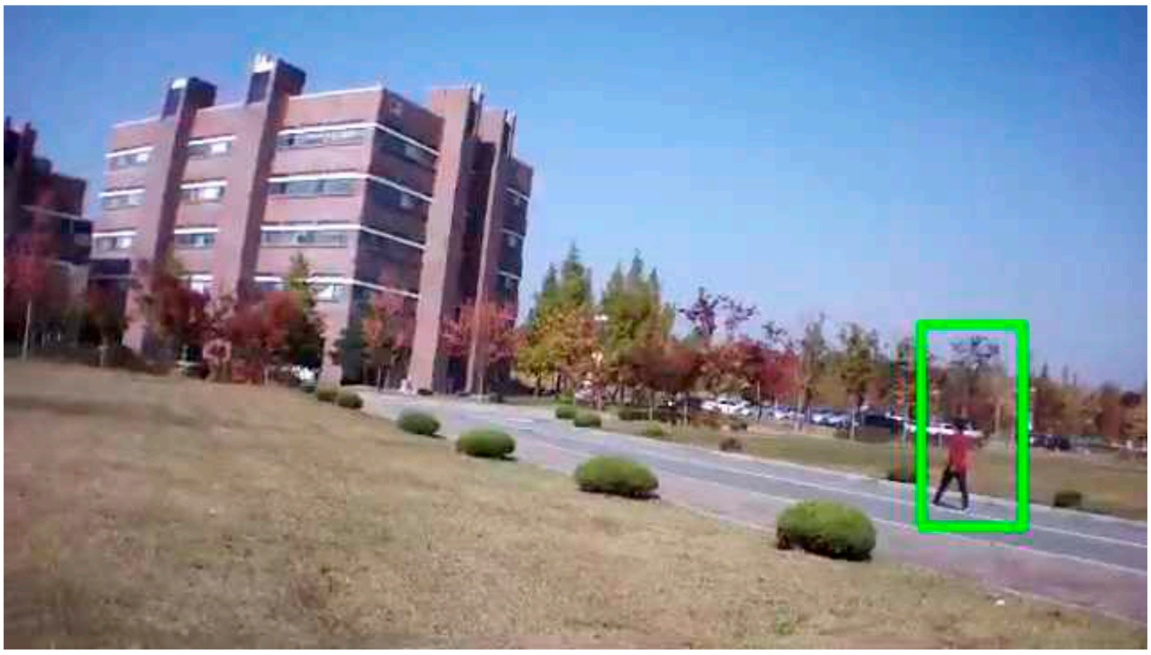

**Figure 21.** Low resolution image of the pedestrian at a far distance.

To mitigate this problem, we can replace the camera on the drone with a higher resolution camera, or to use more sophisticated algorithms to detect our objects.

### *7.4. False Detection of the Pedestrian Detector*

We used a pedestrian detector which is provided by OpenCV 2.4.10, and the accuracy of the pedestrian detector is not within our scope of research. However, the accuracy of the pedestrian detector will affect our algorithm's robustness since the disparity is dependent on the detection performance. Figure 22 shows an example of the false detection of a pedestrian.

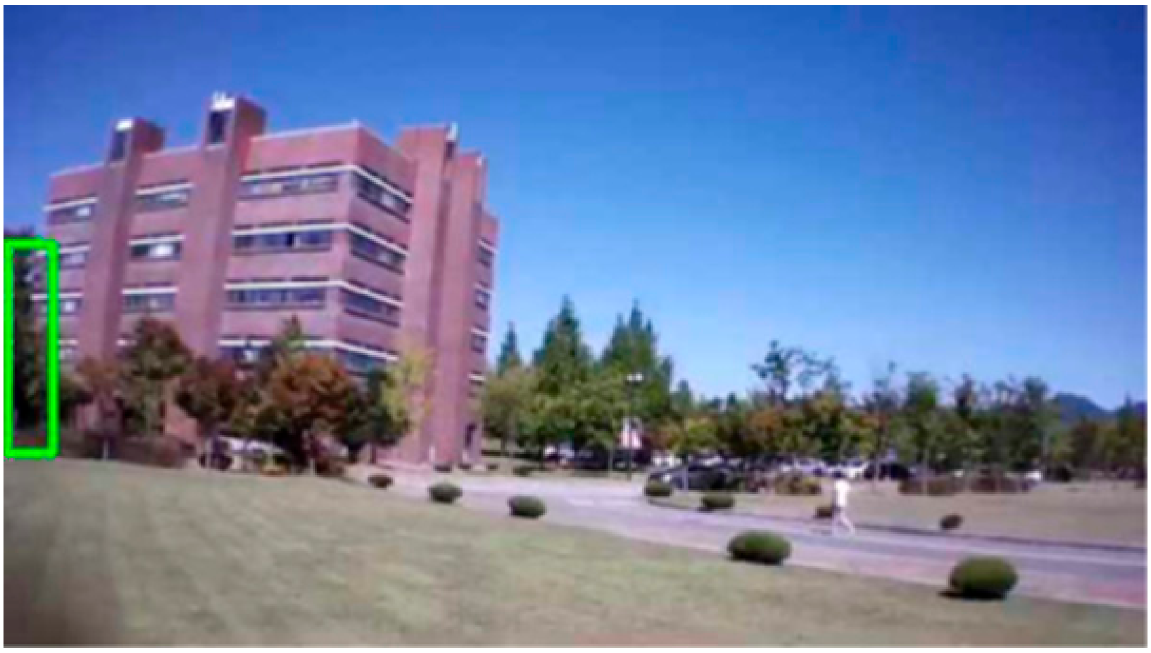

**Figure 22.** False detection of pedestrians.

Our experiments showed that the false detection happens more frequently when the pedestrian is further away than 11 m, or when there are other objects in the background which have long and thin vertical shapes (such as trees, road lights, and pillars). This problem can be mitigated by using more sophisticated pedestrian detectors.

### *7.5. Fast Moving Objects not Moving in a Straight Line*

Our solution works on the assumptions that the drone is moving much faster than the object, and that the drone's camera's frame rate is high. This allows us to model the object's motion as a straight line. For slow-moving objects (e.g., a walking pedestrian), the speed of the drone (11.11 m/s) and the frame rate (200 ms interval) is sufficient for these assumptions to hold true. For fast-moving objects that are not moving in a straight line (e.g., a cheetah chasing a prey, or a car driving in a zig-zag fashion), these assumptions may not be valid and there may be significant errors in the results. However, it is not the natural tendency of such objects to move at such a high speed and in an erratic fashion for long periods of time, and they are likely to revert to slower, linear motion (e.g., when the cheetah gets tired, and the driver of a car at high speed will prefer to drive in a straight line). Since we can repeat our algorithm as often as we like, we will be able to minimize the error and obtain an accurate measurement upon the repeated application of the algorithm.

## 8. Conclusions

This paper proposed a novel system to estimate the location of an object from a single moving camera mounted on a drone. The proposed algorithm aimed to fulfil two objectives: (1) to be able to make the baseline between cameras arbitrarily large, and (2) to be able to remove the depth ambiguity caused by a moving object relative to a moving camera. We have shown in our previous work [16] that, by using three images, we are able to recover the location of an object moving in a direction parallel to the motion of the camera. In this research, we showed that, by using four images, we are able to recover the location of an object moving linearly in an arbitrary direction.

The algorithm works by instructing the drone to fly in a specific pattern with acceleration, which allows us to use a varying set of camera baselines to compute the object position, thus avoiding the depth ambiguity. To maintain generality, we do not assume that we know the class of the object (e.g., person, vehicle, farming equipment, etc.) or the size, speed, and direction of motion of the object. This allows us to apply this technique to various applications, such as locating farm personnel and equipment, mining vehicles, delivery vehicles belonging to logistics supply chains, wild animals in a prairie, etc. Our evaluation results show that our algorithm can achieve a depth-error rate of <8%, and it is equal or better in terms of localization accuracy when compared to other state-of-the-art algorithms.

The proposed algorithm will be very useful for drones with a single camera, including lightweight or low-cost drones. The algorithm can be used in a vast number of applications, such as drone navigation, intelligent CCTV systems, pursuit missions, military missions, etc. For example, this system can be used as a mobile intelligent CCTV system for both indoors and outdoors, and has the ability to locate suspicious entities (such as in supermarkets, factories, private estates, government properties, etc.). Additionally, the algorithm can help in situations when a drone pursues a distant target, such as following a suspected criminal (and avoiding detection by the suspect and avoiding hostile actions on the drone), following endangered animal species (with stealth, avoiding close encounter), and performing military reconnaissance missions (since the system is "passive", it does not emit any signals).

**Author Contributions:** Conceptualization, K.-C.Y.; methodology, K.-C.Y. and I.K.; software, I.K.; validation, I.K.; formal analysis, K.-C.Y. and I.K.; investigation, K.-C.Y. and I.K.; resources, K.-C.Y.; data curation, I.K.; writing—original draft preparation, I.K.; writing—review and editing, K.-C.Y.; visualization, I.K.; supervision, K.-C.Y.; project administration, K.-C.Y.; funding acquisition, K.-C.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1A02019235).

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

## References

1. Hrabar, S.; Sukhatme, G.S. Vision-based navigation through urban canyons. *J. Field Robot.* **2009**, *26*, 431–452. [CrossRef]
2. Benson, E.R.; Reid, J.F.; Zhang, Q. Machine vision-based guidance system for an agricultural small-grain harvester. *Trans. ASAE* **2003**, *46*, 1255–1264. [CrossRef]
3. Lavigne, N.J.; Marshall, J. A landmark-bounded method for large-scale underground mine mapping. *J. Field Robot.* **2012**, *29*, 861–879. [CrossRef]
4. Okatani, T.; Deguchi, K. Shape Reconstruction from an Endoscope Image by Shape from Shading Technique for a Point Light Source at the Projection Center. *Comput. Vis. Image Underst.* **1997**, *66*, 119–131. [CrossRef]
5. Kou, K.; Li, X.; Li, L.; Li, H.; Wu, T. Absolute distance estimation with improved genetic algorithm in laser self-mixing scheme. *Opt. Technol.* **2015**, *68*, 113–119. [CrossRef]
6. Fang, S.-H.; Yang, Y.-H.S. The Impact of Weather Condition on Radio-Based Distance Estimation: A Case Study in GSM Networks with Mobile Measurements. *IEEE Trans. Veh. Technol.* **2016**, *65*, 6444–6453. [CrossRef]
7. Chang, H.; Choi, J.; Kim, M. Experimental research of probabilistic localization of service robots using range image data and indoor GPS system. In Proceedings of the 2006 IEEE Conference on Emerging Technologies and Factory Automation, Prague, Czech Republic, 20–22 September 2006; pp. 1021–1027.
8. Perez-Ramirez, J.; Borah, D.; Voelz, D. Optimal 3-D landmark placement for vehicle localization using heterogeneous sensors. *IEEE Trans. Veh. Technol.* **2013**, *62*, 2987–2999. [CrossRef]
9. Ishihara, M.; Shiina, M.; Suzuki, S. Evaluation of Method of Measuring Distance Between Object and Walls Using Ultrasonic Sensors. *J. Asian Electr. Veh.* **2009**, *7*, 1207–1211. [CrossRef]
10. Majchrzak, J.; Michalski, M.; Wiczynski, G. Distance Estimation with a Long Range Ultrasonic Sensor System. *IEEE Sens. J.* **2009**, *9*, 767–773. [CrossRef]

11. Truong, H.; Abdallah, S.; Rougeaux, S.; Zelinsky, A. A novel mechanism for stereo active vision. In Proceedings of the Australian Conference on Robotics and Automation, Melbourne, Australia, 30 August–4 September 2000.
12. Forsyth, D.; Ponce, J. *Computer Vision: A Modern Approach*; Prentice Hall: Englewood Cliffs, NJ, USA, 2003.
13. Brown, M.Z.; Burschka, O.; Hager, G.D. Advances in computational stereo. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *25*, 993–1008. [CrossRef]
14. Weiss, S.; Scaramuzza, D.; Siegwart, R. Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments. *J. Field Robot.* **2011**, *28*, 854–874. [CrossRef]
15. Jarvis, R. A perspective on range finding techniques for computer vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **1983**, *5*, 122–139. [CrossRef] [PubMed]
16. Kim, I.; Yow, K.C. Object Location Estimation from a Single Flying Camera. In Proceedings of the Ninth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2015), Nice, France, 19–24 July 201.
17. Cigla, C.; Thakker, R.; Matthies, L. Onboard Stereo Vision for Drone Pursuit or Sense and Avoid. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 18–22 June 2018; pp. 738–746.
18. Grewe, L.; Stevenson, G. Drone based user and heading detection using deep learning and stereo vision. In Proceedings of the SPIE 11018, Signal, Processing, Sensor/Information Fusion, and Target, Recognition XXVIII, Baltimore, MD, USA, 15–17 April 2019; p. 110180X. [CrossRef]
19. Nam, D.V.; Gon-Woo, K. Robust Stereo Visual Inertial Navigation System Based on Multi-Stage Outlier Removal in Dynamic Environments. *Sensors* **2020**, *20*, 2922. [CrossRef] [PubMed]
20. Fehrman, B.; McGough, J. Depth Mapping using a Low-Cost Camera Array. In Proceedings of the Southwest Symposium on Image Analysis and Interpretation, San Diego, CA, USA, 6–8 April 2014; pp. 101–104.
21. Wan, D.; Zhou, J. Stereo vision using two PTZ cameras. *Comput. Vis. Image Underst.* **2008**, *112*, 184–194. [CrossRef]
22. Tran, D.X.; Nguyen, T.D.; Bui, H.M.; Nguyen, S.T.; Hoang, H.X.; Pham, T.C.; de Souza-Daw, T. Dual PTZ Cameras Approach for Security Face Detection. In Proceedings of the Fifth IEEE Conference on Communications and Electronics (ICCE), Da Nang, Vietnam, 30 July–1 August 2014; pp. 478–483.
23. Ramírez-Hernández, L.R.; Rodríguez-Quinonez, J.C.; Castro-Toscano, M.J.; Hernández-Balbuena1, D.; Flores-Fuentes, W.; Rascón-Carmona, R.; Lindner, L.; Sergiyenko, O. Improve three-dimensional point localization accuracy in stereo vision systems using a novel camera calibration method. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1–15. [CrossRef]
24. Islam, A.; Asikuzzaman, M.; Khyam, M.O.; Noor-A-Rahim, M.; Pickering, M.R. Stereo Vision-Based 3D Positioning and Tracking. *IEEE Access* **2020**, *8*, 138771–138787. [CrossRef]
25. Kim, J.H.; Kwon, J.W.; Seo, J. Multi-UAV-based stereo vision system without GPS for ground obstacle mapping to assist path planning of UGV. *Electron. Lett.* **2014**, *50*, 1431–1432. [CrossRef]
26. Xu, C.; Yin, C.; Huang, D.; Han, W.; Wang, D. 3D target localization based on multi–unmanned aerial vehicle cooperation. *Meas. Control* **2020**, 1–13. [CrossRef]
27. Achtelik, M.; Bachrach, A.; He, R.; Prentice, S.; Roy, N. Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. In Proceedings of the Unmanned Systems Technology XI. International Society for Optics and Photonics, Orlando, FL, USA, 4–17 April 2009; Volume 7332. [CrossRef]
28. Santamaria-Navarro, A.; Loianno, G.; Solà, J.; Kumar, V.; Andrade-Cetto, J. Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors. *Auton. Robot.* **2018**, *42*, 1263–1280.
29. Chen, I.H.; Wang, S.J. An Efficient Approach for the Calibration of Multiple PTZ Cameras. *Autom. Sci. Eng.* **2007**, *4*, 286–293. [CrossRef]
30. Zhang, R.; Liu, H.H.T. Vision-Based Relative Altitude Estimation of Small Unmanned Aerial Vehicles in Target Localization. In Proceedings of the American Control Conference, San Francisco, CA, USA, 29 June–1 July 2011; pp. 4622–4627.
31. Sereewattana, M.; Ruchanurucks, M.; Siddhichai, S. Depth Estimation of Markers for UAV Automatic Landing Control Using Stereo Vision with a Single Camera. In Proceedings of the International Conference on Information and Communication Technology for Embedded Systems, (ICICTES), Ayutthaya, Thailand, 23–25 January 2014.

32. Sereewattana, M.; Ruchanurucks, M.; Rakprayoon, P.; Siddhichai, S.; Hasegawa, S. Automatic Landing for Fixed-wing UAV Using Stereo Vision with A Single Camera and An Oriented Sensor: A Concept. In Proceedings of the 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Busan, Korea, 7–11 July 2015; pp. 29–34.
33. Nguyen, P.H.; Kim, K.W.; Lee, Y.W.; Park, K.R. Remote Marker-Based Tracking for UAV Landing Using Visible-Light Camera Sensor. *Sensors* **2017**, *17*, 1987. [CrossRef] [PubMed]
34. Huang, R.; Tan, P.; Chen, B.M. Monocular Vision-Based Autonomous Navigation System on a Toy Quadcopter in Unknown Environments. In Proceedings of the International Conference on Unmanned Aircraft Systems, Denver, CO, USA, 9–12 June 2015; pp. 1260–1269.
35. Bian, J.; Hui, X.; Zhao, X.; Tan, M. A monocular vision–based perception approach for unmanned aerial vehicle close proximity transmission tower inspection. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1–20. [CrossRef]
36. Parrot Rolling Spider. Available online: http://www.parrot.com/usa/products/rolling-spider/ (accessed on 9 June 2020).
37. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
38. Kalal, Z.; Mikolajczyk, K.; Matas, J. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 1409–1422. [CrossRef] [PubMed]
39. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 583–596. [CrossRef] [PubMed]
40. Parrot AR.Drone2.0: Technical Specifications. Available online: http://ardrone2.parrot.com/ardrone-2/specifications/ (accessed on 9 June 2020).
41. CV Drone: OpenCV + AR.Drone. Available online: https://github.com/puku0x/cvdrone/ (accessed on 9 June 2020).