Object Location Estimation from a Single Flying Camera

Insu Kim and Kin Choong Yow
GIST College, Gwangju Institute of Science and Technology
Gwangju, Republic of Korea
e-mail: ahinsutime@gist.ac.kr, kcyow@gist.ac.kr

***Abstract*—With the recent popularity and ubiquity of drones, there had been an increasing demand to deploy drones for the detection, localization and tracking of objects in a scene (e.g., pedestrians, cars, etc.). The problem with a single camera drone is that it is impossible to estimate distances from a single image. Although the drone can fly to another position to take a second image, the object that we are tracking may have moved during that time interval, rendering traditional stereo-vision algorithms useless. In this paper, we propose a novel system that instructs the drone to fly in a specific pattern so as to achieve a large baseline, and use three images (instead of the traditional two) to recover the distance to the object that is moving. The experimental results show that our algorithm can estimate depth with better or equal accuracy than other state-of-the-art methods. This algorithm would have great significance for small or low cost drones which are unable to carry additional devices (apart from the built-in camera), thus enhancing their ubiquity of use.**

Keywords - computer vision; drone; stereo-vision; distance extraction; object detection; location mapping.

I. INTRODUCTION

With the recent popularity and ubiquity of drones or Unmanned Aerial Vehicles (UAV), they have been increasingly deployed in various tasks such as object localization and tracking. Drones often carry a high-definition camera and it can be used for surveillance, expedition guidance, search and rescue, etc. However, with a single image of an object, it is usually impossible to obtain the distance of the object from the camera because we usually do not know the size of the object.

Traditionally, such a problem can be solved with stereo-vision, i.e., putting a second camera some distance away and computing the disparity of the object in the two images. However, existing stereo-vision algorithms require that the distance between the two cameras (i.e., the baseline) be fixed because the image disparity is a function of the object distance and the baseline.

Drones can be and have been fitted with two cameras for their mission. Carrillo et al. [1] showed the possibility of using stereo vision with inertial navigation system which can estimate the UAV's position accurately. They used two separated fixed camera on the drone. Knoppe [2] also proposed a system for a drone carrying a stereo camera to get ground surface scanning data. Schauwecker and Zell [3] introduced more sophisticated method for navigating Micro Aerial Vehicles (MAV) using four cameras. They performed stereo matching separately for the downward and forward of the MAV.

Certainly, the use of additional cameras could generate problems for a drone such as increased payload, insufficient power, reduced duration of flight, instability, etc. However, a more important problem is that the baseline between the two cameras is short relative to the distance of the object, resulting in very large errors in distance estimation. Since the drone can fly to any location with little or no restrictions, the obvious solution is to carry a single camera and fly as far as the environment allows (i.e., without losing sight of the object or crashing into an obstacle) so as to maximize the baseline. Traditional stereo-vision algorithms will still work, and it does not matter whether the two images are taken from two separate cameras, or from a single camera that had moved. This, however, works only if the target object remains stationary between the two views.

Hence, in this paper, we propose a novel algorithm that enables a drone carrying a single camera to produce an accurate estimate of the distance of a stationary or moving object. Our algorithm works by instructing the drone to fly in a specific pattern so as to achieve a large baseline, and use *three* images (instead of the traditional two) to recover the distance to the (moving) object.

Our proposed system is very effective for small or low cost drones (e.g., Parrot rolling spider [4]) which are unable to carry additional devices (apart from the built-in camera). This increases the ubiquity of using drones for object localization and tracking. Once the distance of the object from the drone is known (and angle, which can be obtained from a magnetic compass), we can map the object location to any global coordinate system (assuming that we know the drone's position e.g., through Global Positioning System (GPS)).

The rest of the paper is organized as follows: Section II discusses some related work, and Section III describes the proposed algorithm. Section IV discusses the implementation details and Section V provides the experimental results. Section VI concludes the paper.

II. RELATED WORK

Similar work has been done by Zhang and Liu [5]. They 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. In our system, we do not make the assumption that we know the size of the object. This means that our algorithm is more generic and can be extended to locate and track any kind of objects. Sereewattana et al. [6]

 ISBN: 978-1-61208-418-3

did depth estimation of color markers for automatic landing control of UAV 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 meters). Kendall et al. [7] also proposed novel system for UAV which tracking an object of known color and size to get the depth information. For our case, our objects can be any size, moving and can be arbitrarily far (limited by the accuracy of the object detection algorithm).

A moving camera can also be compared to a Pan-Tilt-Zoom (PTZ) camera. Wan and Zhou [8] introduced a novel stereo rectification method for a dual-fixed-PTZ-camera system that can recover distance accurately. However, their PTZ camera pair is fixed so the object may disappear from the cameras' view. In comparison, our system can fly to different positions to avoid occlusion, as well as choosing the length of the baseline. Tran et al. [9] proposed a system that performed face detection using dual fixed PTZ cameras for large area security system. They showed good result for the detection rate (99.92%) with indoor detection range (5 to 20m). However, their work is concerned about object recognition rather than to estimate the distance of the object from the camera.

III. PROPOSED ALGORITHM

A. System Overview

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 and angle θ_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 Inertial Navigation System (INS), if available).

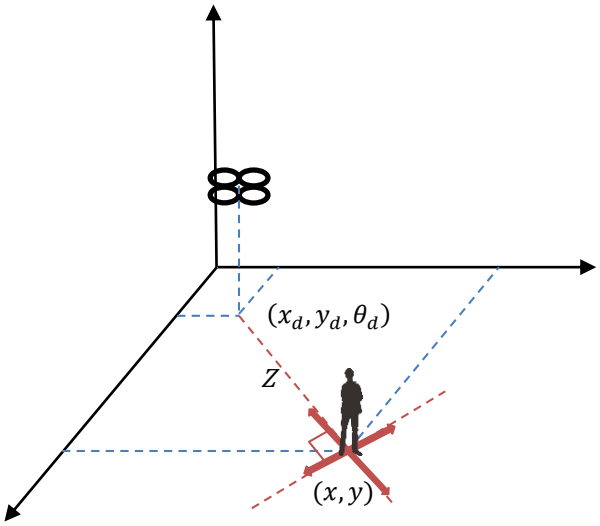

Figure 1. Overview of the localization system

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. This process can be repeated as often as necessary. Figure 2 gives a flowchart of the system.

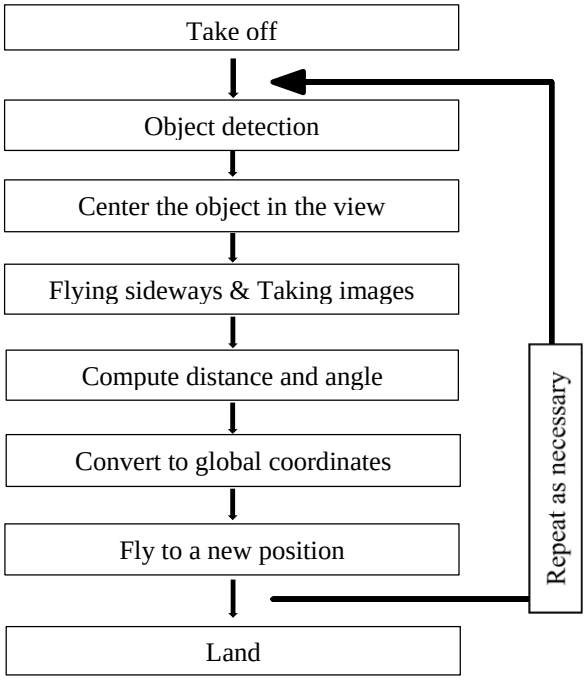

Figure 2. System flowchart

B. Object detection

Our system is not restricted to work on only one class of objects. It can be extended 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 3 shows an example of the pedestrian detection algorithm in OpenCV, which displays a bounding box over the detected pedestrian.

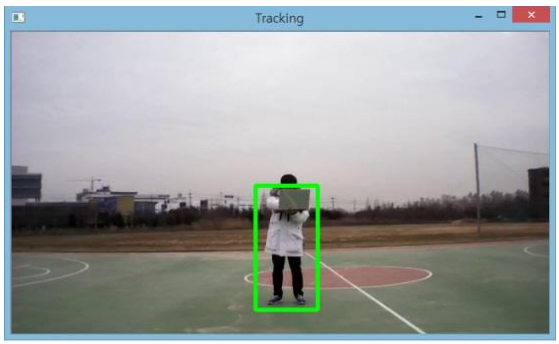

Figure 3. Example of pedestrian detection from the drone

The pedestrian detector from OpenCV can detect more than one person in a scene. Like with any other tracking algorithm, we need to make use of additional information (e.g., color, size, motion vectors, etc.) to correctly match the object in different scenes. However, our experiment shows that if the pedestrian is further than 11m from the drone's camera, the false detection rate becomes significantly increased. Thus in our experiments, we restrict our study to detect pedestrians at depths below 11 meters. The design of a more accurate pedestrian detector is not in the scope of our work.

C. Camera Calibration

In order to obtain an accurate estimation of the depth of an object, the camera needs to be calibrated. The basic concept of the classical structure-from-motion algorithm is from the geometry shown in Figure 4.

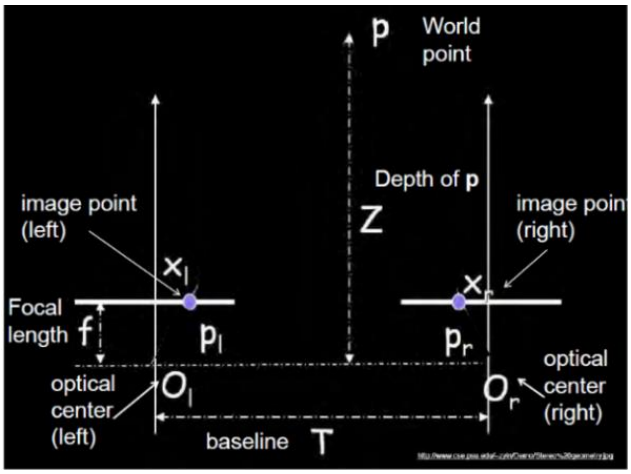

Figure 4. Classical disparity formula diagram

From Figure 4, using similar triangles, the following relation for depth, Z, can be derived easily:

$$Z = f\frac{T}{x_r - x_l} \quad (1)$$

where f is the focal length, T is the baseline, and $x_r - x_l$ is the image disparity. The camera needs to be calibrated to find f in order to be able to obtain Z based on the image disparity.

However, in our formulation (discussed in the next section), we need another important parameter which maps the image offset p (position of the object in the image from the image center) to the angle θ subtended by the object from the image center (see Figure 5). As the image offset p is dependent entirely on the focal length, we choose to calibrate for θ/p instead of the focal length f.

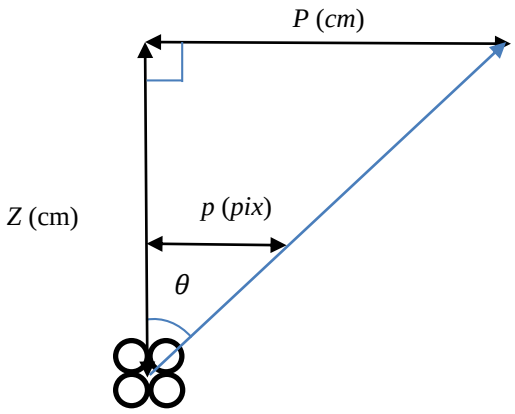

Figure 5. Geometry of view angle calibration

We can obtain this value by placing an object at various positions in the drone's field of view (Figure 6) and measuring the image offset p and the angle subtended θ.

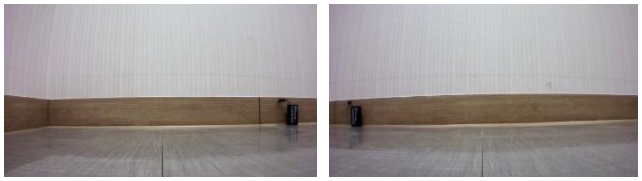

Figure 6. Images of an object taken at various positions

Table I shows the data measured from Figure 6. While the θ/p ratio is not constant throughout the entire image, the variation was found to be quite small (0.001°/pixel) between the two cases where the object is in the image center and at the image edge.

TABLE I. CALIBRATION OF THE DRONE'S CAMERA

Z (cm)	*P (cm)*	*arctan(Z/P) (°)*	*p (pixels)*	*θ/p (°/pixels)*
92	*57*	*31.78°*	*320*	*0.099*

At the image edge of 320 pixels (for a 640x480 pixel image), the θ/p ratio is found to be 0.099°/pixels.

D. Baseline calibration

Another important parameter to calibrate is the distance between the drone positions at each successive image capture. This represents the baseline between the images. To obtain these distances, the drone is instructed to take a sequence of images at constant rate while flying with a preset speed to the left (or right). To reduce the error in estimating depth, the length of the baseline should be as long as possible.

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 (we choose 'left' for the ease of discussion) at the maximum speed (i.e., roll angle phi = -1.0, normalized), while taking images at 500ms interval for a total of 11 images. These

 ISBN: 978-1-61208-418-3

values were chosen after numerous tries to give the best tradeoff between distance flown and the time taken to complete, as the object (pedestrian) may have moved during that time interval.

To measure the length of the baselines, we placed numerous markers on a wall, and made the drone fly parallel to the wall, taking images as mentioned above. From the images, we can determine the positions of the drone where the images were taken. We observe that the first image (i.e., at position 0) was exactly the same as the image at hovering (i.e., it was taken before the drone has even started to rotate (roll)), and the second image (i.e., at position 1) was taken just after the drone has completed its rotation (roll) to the left (but before any horizontal translation takes place – so no change in its position). The rest of the images (at positions 2 to 10) were of increasing distances between each other, which is due to the inertial and the acceleration of the drone, which needed some time to accelerate to the desired speed.

We repeated the experiment three times, and averaged the results of the four experiments to produce the baselines shown in Table II.

TABLE II. ELINE CALIBRATION

position	*Average distance from previous position (cm)*
0	*- (Hover)*
1	*0 (Rotate)*
2	*47.50*
3	*51.75*
4	*75.75*
5	*91.25*
6	*100.5*
7	*108.75*
8	*126.75*
9	*146.50*
10	*189.25*

The baseline between any two images is simply the sum of the values between them. For example, the baseline between image 0 and image 5 is 47.50+51.75+75.75+91.25 = 266.25(cm).

E. Depth estimation

We now present our formulation for estimating the depth of an object from the drone's camera images and its position. We will divide our discussion into 3 parts (1) stationary object, (2) moving object in a direction parallel to the drone's flight, and (3) moving object in a direction perpendicular to the drone's flight. For a moving object in an arbitrary direction, a similar analysis has been performed but the result is not shown here due to the lack of space.

1) Stationary Object

If the object is stationary, then our problem reduces itself to the classical case of stereo-vision. Figure 7 illustrates the geometry needed for the computation of depth *Z* in this case.

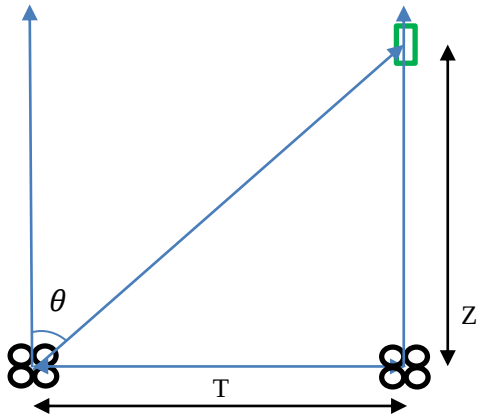

Figure 7. Classical stereo vision method for the stationary pedestrian

We will first instruct the drone to takeoff, and then hover for a few seconds for it to stabilize itself. Then, we will call the object detector function to find any objects within its view. For the ease of discussion, let us use the example of pedestrian detection. After the drone is stabilized, we call the HOG pedestrian detector from the OpenCV library. If more than one pedestrian is found, we need to choose which pedestrian is the one that we are trying to localize. After we have found the pedestrian, we rotate the drone (yaw) so that the pedestrian (i.e., the centroid of the bounding box) is in the center of the image (see Figure 2 for an overview of the system).

The next step is the most important step of the algorithm. The drone is instructed to fly left and take 11 images using the same parameters as in the baseline calibration. After the images are taken, the drone will return to its hovering state and then examine whether the pedestrian is detected in each image. For example, if the pedestrian disappears at image 6, then the drone will use image 0 and image 5 to calculate the baseline (otherwise use the last image so as to obtain the largest baseline). From the image offset *p* of the pedestrian in image 5, we can also use our calibrated θ/p to find the angle θ at image 5.

From Figure 7 we can obtain the equation:

$$Z = \frac{T}{\tan\theta} \qquad (2)$$

from which we can compute the depth *Z*.

2) Object moving parallel to drone's flight

Here, we assume that the object (pedestrian) is moving at a constant speed in a straight line. If he is not, we can approximate the pedestrian movement as piecewise linear. The loop in Figure 2 needs to be repeated for more accurate localization results. Figure 8 illustrates the geometry needed for the computation of depth *Z* in this case.

 ISBN: 978-1-61208-418-3

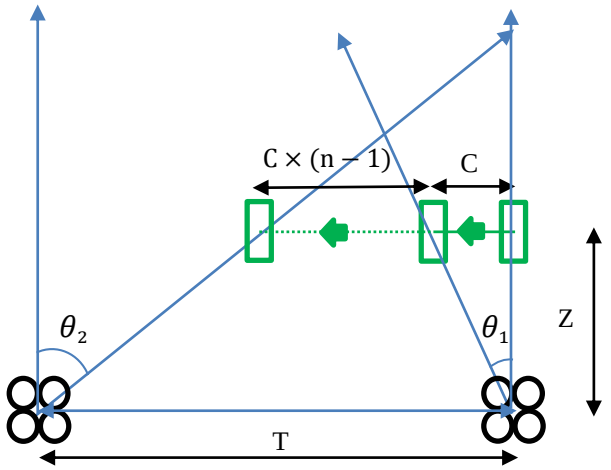

Figure 8. Geometry for pedestrian moving parallel to the baseline

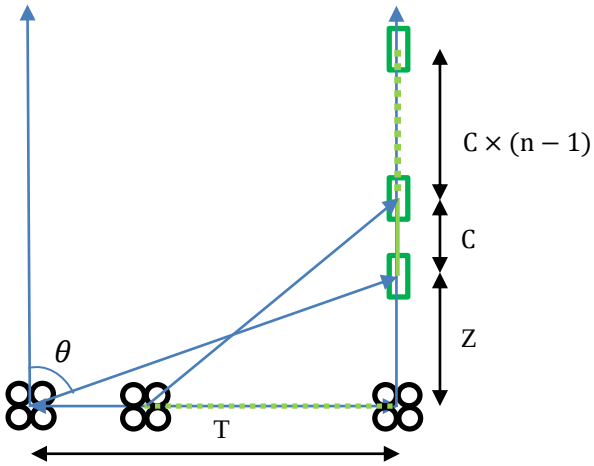

Figure 9. Geometry for pedestrian moving perpendicular to the baseline

The key idea here is that we now need **three** images, at position 0, 1 and *n* (the last image where the pedestrian can still be seen) to compute the depth *Z*. Due to the fact that at position 1 the drone has just completed its rotation (roll) to the left, the drone has not moved yet but the pedestrian had moved by a distance C. This gives us the very important image offset p_1 that allows us to calculate θ_1. At the *n*th position, at angle θ_2, we know that the pedestrian has moved by an additional distance of $C \times (n-1)$. So from the geometry, we can generate the following relation:

$$Z\tan\theta_2 = T - C \times n \quad (3)$$

$$C = Z\tan\theta_1 \quad (4)$$

Eliminating 'C', we obtain the following equation.

$$Z = \frac{T}{(tan\theta_2)+n(tan\theta_1)} \quad (5)$$

from which we obtain the depth Z.

If the pedestrian moves to the right instead, equation (5) is still valid. In that case, the pedestrian may disappear from the image much sooner. To overcome this, we can simply make the drone fly to the right for a second computation.

3) Object moving perpendicular to drone's flight

In the case of a pedestrian moving perpendicular to the baseline of the drone, the computation is very simple. Figure 9 illustrates the geometry needed for the computation of depth Z in this case. Assuming that the pedestrian starts from a distance of $C \times n + Z$ from position 0, after n images, he would be at a distance Z away from position 0. This is the same as the stationary case and we simply need equation (2) to obtain the depth *Z*.

F. Transformation to global coordinates

After we have obtained the depth *Z* and angle θ, we can use them together with the drone position to calculate the global coordinates of the pedestrian. Figure 10 shows the geometry for calculating the coordinate transformation for each of the 3 cases.

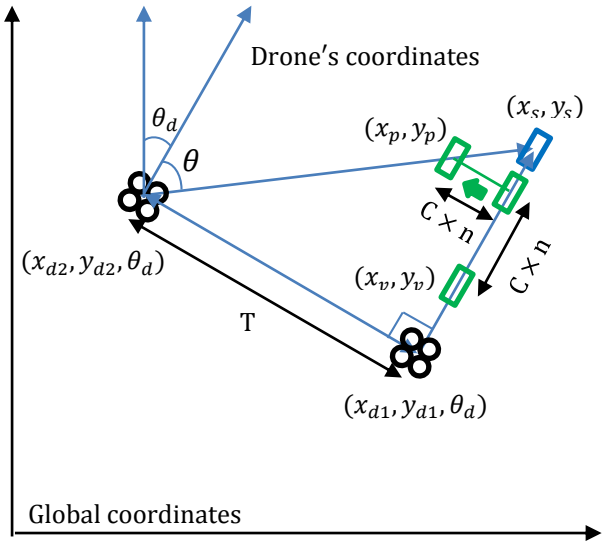

Figure 10. Coordinate transformation

In Figure 10, $(x_{d2}, y_{d2}, \theta_d)$ *and* $(x_{d1}, y_{d1}, \theta_d)$ refers the positions of the drone (which is assumed to be known). $(x_s, y_s), (x_p, y_p), (x_v, y_v)$ are the positions of the pedestrian in each of the three cases of stationary, moving parallel, or moving perpendicular (vertical) to the baseline, respectively. Note that the initial coordinate of the drone $(x_{d1}, y_{d1}, \theta_d)$ is vital which is used as boundary condition for coordinate transformation. From this geometry, we can directly

calculate the position of a pedestrian's position using one of the three following equations.

$$(x_s, y_s) = (x_{d1} + \frac{\tan\theta}{T}\sin\theta_d\ , y_{d1} + \frac{\tan\theta}{T}\cos\theta_d) \quad (6)$$

$$(x_p, y_p) = (x_{d1} + (T - \frac{T(tan\theta)}{(tan\theta_2)+n(tan\theta_1)})\cos\theta_d,$$
$$y_{d1} + (T - \frac{T(tan\theta)}{(tan\theta_2)+n(tan\theta_1)})\sin\theta_d) \quad (7)$$

$$(x_v, y_v) = (x_{d1} + (\frac{\tan\theta}{T} - \mathrm{C} \times \mathrm{n})\sin\theta_d,$$
$$y_{d1} + (\frac{\tan\theta}{T} - \mathrm{C} \times \mathrm{n})\cos\theta_d) \quad (8)$$

IV. IMPLEMENTATION

A. Hardware

The drone used in this study is the Parrot AR.Drone2 GPS edition [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 11. Parrot AR.Drone2.0

B. Software

The software we used to control the drone is the "CV Drone" package which is available from Github [11]. The image processing routines were from the OpenCV 2.4.10 library, and the entire system was developed in Microsoft Visual Studio 2013.

V. EXPERMENTAL RESULTS

A. Experiments

To evaluate our proposed algorithm, we conduct experiments for each of the three cases of stationary, moving parallel and moving perpendicular pedestrians. For each case, we perform the experiment four times with the pedestrian at 2.75, 5.5, 7.25, and 9 meters from the drone. As the HOG pedestrian detection from the OpenCV library did not work well from 11 meters and beyond, we stopped the experiment at 9 meters.

For the moving cases, the pedestrian was asked to move in the required direction at a speed of 0.6 m/s, covering a distance of about 3m in the 11 images that was captured. Examples of the images captured by the drone (at 5.5m, stationary) are shown in Figure 12. Notice that between the first and second image, there is only a rotation of the drone and there is no horizontal movement.

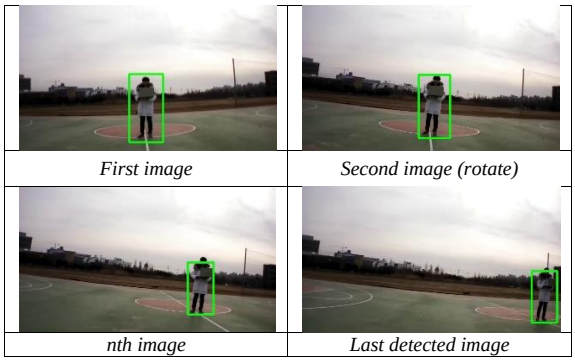

Figure 12. Examples of Image Sequence and pedestrian detection (5.5m, stationary pedestrian)

For each of the experiments, we compute depth error

$$\text{Depth error } (\%) = |Z - Z'|/Z \quad (9)$$

where Z refers to the actual depth while Z' indicates measured depth. Also, since we compute the pedestrian position in global coordinates, we also compute the position error using the following equation:

$$\text{Position error } (\%) = (\sqrt{(x - x')^2 + (y - y')^2})/Z \quad (10)$$

Here, (x, y) refers the actual position of a pedestrian while $(x',\ y')$ indicates the measured position. The results are shown in Tables III to V.

TABLE III. STATIONARY PEDESTRIAN

Actual depth, Z(m)	*Measured depth, Z'(m)*	*Depth error rate (%)*	*Position error rate (%)*
2.75	*2.35*	*14.5*	*16.5*
5.50	*6.01*	*9.3*	*18.0*
7.25	*7.51*	*3.6*	*12.2*
9.00	*8.96*	*0.4*	*7.2*

TABLE IV. PERPENDICULARLY MOVING PEDESTRIAN

Actual depth, Z(m)	*Measured depth, Z'(m)*	*Depth error rate (%)*	*Position error rate (%)*
4.50	*4.08*	*9.3*	*9.4*
7.60	*6.69*	*12.0*	*12.9*
5.45	*5.33*	*2.2*	*5.86*
6.90	*6.75*	*2.2*	*2.2*

TABLE V. PARALLELLY MOVING PEDESTRIANS

Actual depth, Z(m)	*Measured depth, Z'(m)*	*Depth error rate (%)*	*Position error rate (%)*
2.75	*3.43*	*24.7*	*27.7*
5.50	*6.20*	*12.3*	*12.7*
7.25	*8.28*	*14.2*	*22.8*
9.00	*9.82*	*9.1*	*11.7*

 ISBN: 978-1-61208-418-3

We can see from the tables that with the exception for the parallel case at the nearest distance (i.e., 2.75), all the experiments yield good results at < 15% depth error rate.

B. Comparison with other techniques

In this section, we compare our algorithm with the classical stereo-vision technique of computing image disparity and then calculating the distance using equation (1). Table VI shows the results.

TABLE VI. 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 short depth (2.75m), it actually works better than the classical method for the larger depths. We further compare our method with the method proposed by Sereewattana et al. [6], which is only evaluated for stationary objects less than 3m in depth. Table VII shows the results.

TABLE VII. COMPARISON WITH OTHER SYSTEMS (STATIONARY CASE)

System	***Accuracy (%)***
M. Sereewattana et al. [6] (Only for stationary object below 3m)	*3.9 ~ 12.4*
Classical stereo vision depth extraction	*11.0 ~ 31.4*
Our system	*0.4 ~ 14.5*

The results showed that our algorithm can estimate depth with better or equal accuracy than other state-of-the-art methods.

VI. CONCLUSION AND FUTURE WORKS

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 instructs the drone to fly in a specific pattern, which allows us to estimate the baselines between images so as to obtain depth. The algorithm is not restricted to any particular class of objects and can be easily extended to any class of objects. In addition, our formulation makes the novel use of **three** images, which allows us to extract depth even when the object is moving (with the assumption of constant speed and in a straight line). Experiments showed that our algorithm can estimate depth with better or equal accuracy than other state-of-the-art methods.

In this paper, we only reported the analysis and results of a pedestrian moving either parallel or perpendicular to the drone's flight, due to the lack of space. We already have the analysis of a pedestrian moving in an arbitrary direction, and our future work will be directed to complete the experiments for it. In addition, we will expand the capability of the system to cope with non-linear and non-constant pedestrian motion, and also occlusion (e.g., the pedestrian had turned round the corner of a building).

REFERENCES

[1] L. R. G. Carrillo, A. E. D. Lopez, R. Lozno, and C. pegard, "Combining Stereo Vision and Inertial Navigation System for a Quad-Rotor UAV," Journal of Intelligent & Robotic Systems, vol. 65(1-4), 2011, pp. 373-387.

[2] K. Knoppe, "A Lightweight Digital Stereoscopic Camera System," Institute for Geoinformatics Unsiversity of Munster, Matriculation No. 341901, 2013, pp. 1-58.

[3] K. Schauwecker and A. Zell, "On-Board Dual-Stereo-Vision for the Navigation of an Autonomous MAV," Journal of Intelligent & Robotic Systems, 2014, pp. 1-16.

[4] Parrot Rolling Spider. [Online]. Retreived from: http://www.parrot.com/usa/products/rolling-spider/ on 2015. 6. 9.

[5] R. Zhang and H. H. T. Liu, "Vision-Based Relative Altitude Estimation of Small Unmanned Aerial Vehicles in Target Localization," American Control Conference, June. 2011, pp. 4622-4627, ISBN: 978-1-4577-0080-4

[6] M. Sereewattana, M. Ruchanurucks, and S. Siddhichai, "Depth Estimation of Markers for UAV Automatic Landing Control Using Stereo Vision with a Single Camera," ICICTES, 2014.

[7] A. G. Kendall, N. N. Salvapantula, and K. A. Stol, "On-Board Object Tracking Control of a Quadcopter with Monocular Vision," International Conference on Unmanned Aircraft Systems, 2014, pp. 404-411.

[8] D. Wan and J. Zhou, "Stereo vision using to PTZ cameras," Computer Vision and Image Understanding, vol. 112(2), 2008, pp. 184–194.

[9] D. X. Tran et al, "Dual PTZ Cameras Approach for Security Face Dectection," Communications and Electronics, July. 2014, pp. 478-483, ISBN: 978-1-4799-5049-2

[10] Parrot AR.Drone2.0: Technical specifications. [Online]. Retrieved from: http://ardrone2.parrot.com/ardrone-2/specifications/ on 2015. 6. 9.

[11] CV Drone: OpenCV + AR.Drone. [Online]. Retrieved from: https://github.com/puku0x/cvdrone/ on 2015. 6. 9.

 ISBN: 978-1-61208-418-3