Vision-Based Obstacle Avoidance for Small Drone using Monocular Camera

To cite this article: Retaj Raheem Nhair and Tawfiq A. Al-Assadi 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **928** 032048

View the article online for updates and enhancements.

Vision-Based Obstacle Avoidance for Small Drone using Monocular Camera

Retaj Raheem Nhair[1], Tawfiq A. Al-Assadi[2]

[1,2]College of Information Technology, University of Babylon, Babylon, Iraq
[1]E-mail: retaj.alatabi@student.uobabylon.edu.iq
[2]E-mail: tawfiqasadi63@yahoo.com

Abstract

This paper proposes an obstacle avoidance technique for small size drones by using a monocular camera. This method can assist the drone for doing their mission and avoid hitting any obstacle or minimise the collision as possible. For avoiding an obstacle while the drone is doing its mission, We use a computer vision technique to determines the free zone along with the obstacle that may cause a crash and send feedback control to the drone. The small size drone receives feedback about the obstacle and commands the drone to moves to a safe area then resume its trajectory.

Keywords: *obstacle avoidance, trajectory follows, monocular camera, computer vision.*

1. Introduction

During the last decade, the increase in computing efficiency and the developments in microelectronics make commercial micro aerial vehicles (MAVs) widely used in many applications. Aerial photography, surveillance, automated inspection, and even delivery of goods are practical application use the drone to do its missions. The critical component for the success of any small size drone is the ability to automatically obstacle detection and avoidance. The commercially available drones propose that the trajectory credited with the user is collision-free or supports only limited obstacle avoidance capabilities. To avoid breaking the drone while it was accomplishing its tasks, an avoiding method that prevents the collision is required.

Trajectory generation mainly depends on the application and assumptions about the environment that might be a problem faced by the drone because it required to navigate in various environments.

The topic "Obstacle Avoidance" getting the attention of many researchers; most of the researches pointed in two directions: the first direction of obstacle avoidance depends on non-visual based. The non-visual processes mainly depend on the sense of the environments using sensors such as use Planar laser range finders that use LIDAR technologies [1] or RADAR for detecting obstacle and specify its distance from the drone. Although These types of equipment are a powerful tool to recognise the surrounding objects, it heavy relative to the payload of the small size drone and provides incomplete data for three-dimensional obstacle detection. Other functional range finders, such as

Microsoft Kinect [2], currently do not operate outdoors because the sunlight interference effect readings, so it should work only for indoor duties.

The second direction of obstacle avoidance depends on using a camera (vision-based) to recognise the object that in front of the drone. The vision-based are developed and implemented following monocular [3][4], stereo vision methodologies [5][6], or even motion parallax, each of them having advantages and disadvantages depending on the application area of the drone. The Motion parallax is not suitable for MAVs since they require massive Computation to work well. The earlier research gets a better sense of the environment by suggesting of combines using both the vision-based and the sensors [7][8][9] together, which should operate independently of each other in unison and get the detection systems with minimum errors. Use the MAVs has many limitations; it should use equipment with low weight for detecting an obstacle. In this work, we use the low cost, low power consumption, and lightweight web camera that captures an image in real-time while the drone is flying that provides a ready-to-run solution.

The remainder of this paper organised as follows; Section 2 describes the related work. Section 3 shows the required components that we used in our work. Section 4 explains the proposed system that we used to detect an object and command the drone for ignoring crashes. Section 5 reports experiment ts with a real flying drone (a commercial Drone F550 hexacopter are used). Section 6 shows the conclusions and futures works.

2. Related Work

Nowadays, available commercial MAVs use GPS sensors to perform an autonomous flight by using waypoints navigation. It is useful for much MAVs applications that can do its mission autonomously, but the challenge is that it cannot deal with an obstacle during its mission that can cause damage to the drone. There are different collision avoidance techniques can be applied to MAV for achieving autonomous navigation. Various techniques used reactive approach, while the other use full planning based approaches [10]. The reactive approach focuses on taking the reading from the sensor and instantly creating motion control. On the other hand, the full planning-based approach needs to build temporary representation. The benefits of using a reactive approach are fast responding and less computationally.

Many researchers using the vision-based method rather than using the sensor-based method because the camera can sense for long-range with less weight and low power consumption. Aguilar et al. in [11] use feature point detector Speeded Up Robust Features (SURF), For detecting obstacles, they compare the image obtained from an onboard camera that captures in real-time with a database of obstacles.

Mori and Scherer [12] proposed an approach to use SURF feature matches in combination with template matching to compare relative obstacle sizes with different image spacing, that can see change size change in consecutive image that captures to predict the time collision.

Lee et al. [13] Refer to combined both Scale Invariant Feature Transform (SIFT) descriptor and multi-scale-oriented-patches (MOPS) to show 3D information of the obstacles. In which, the MOPS extract the edges and corners of the object, then use SIFT to detect the internal outline information.

Benn et al. [14] presented an approach based on using colour segmentation to distinctly separate free space from the obstacles and uses edge detection to distinguish an obstacle. They use the "Canny" method for edge detection due to its benefit in little discontinuity and generating sharp edges. They applied this technique for indoor navigation.

In additional, Roberts et al. [15] developed an approach that uses line segmentation and optical flow for detecting trees. However, their method works entirely for the forest environment.

In this work, we use an onboard camera for detecting frontal objects and prevent the drone from making a collision with any object.

3. Requirement Components

The benefit of using multi-rotor is the ability to hovering in the air and stay at the same location if required. We suggest to developed algorithm works directly on the small computer that carries by the drone to reduces the delay time between each frame capture by drone onboard camera and the Ground Control Station (GCS). GCS receives the image from the drone and uses computer vision to recognise any obstacles.

This method applied to a hexacopter vehicle use frame F550; we use the Pixhawk as the main controller board that responsible for low-level control of the multi-rotor. The main control board includes a gyroscope, accelerometers, magnetometer, and barometer also can connect to an external GPS module. The Pixhawk use Ardupilot firmware [16] as flight controller software that is free software has many features as the ability to communicate with another device using the MAVLink Communication protocol [17], which are developed especially for the drone applications.

The autonomous control of the drone can accomplish by performing the command algorithm in a small portable computer such as Raspberry Pi [18]. Raspberry Pi is a small size and lightweight single-board computer that can be carried by the drone for communicating directly with the Pixhawk by the MAVLink protocols. MAVLink allows for bi-directional communication between GCS and the drone (both can send and receive MAVLink message), which can read the drone sensors such as get the current position from GPS or can send control movement to the drone.

The drone uses a web camera that installs in the frontal of the drone (as shown in Figure 1). In this experiment, we use Raspbian Buster Desktop as the primary operating system (OS) that based on Linux for the Raspberry Pi4 and use Python language for writing code for the OpenCV 4.1.0 open source graphic library [19].

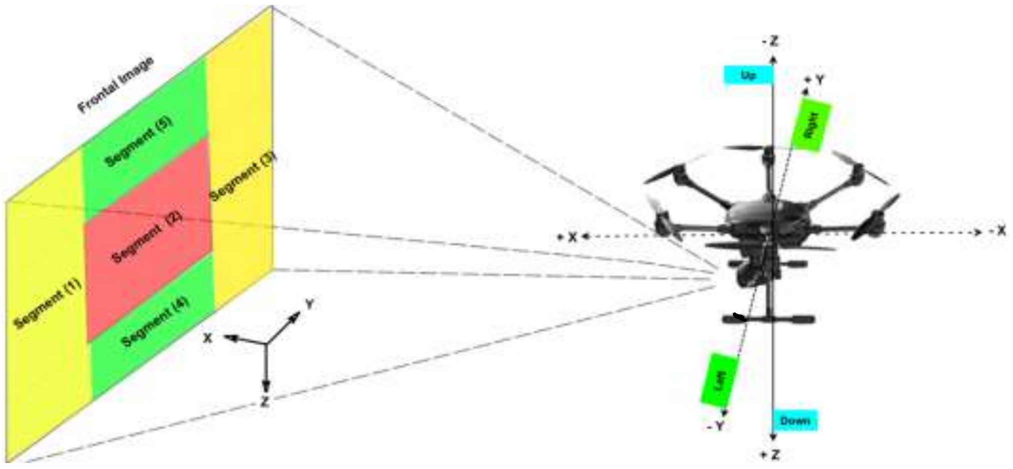

Figure 1. Frontal Camera Image.

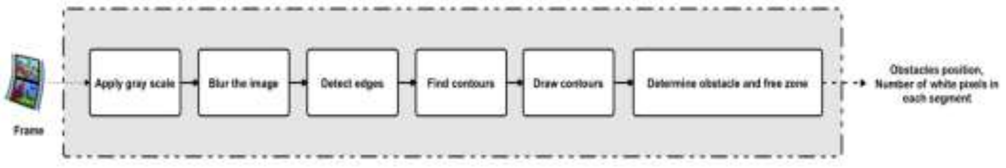
Frame
Apply gray scale
Blur the image
Detect edges
Find contours
Draw contours
Determine obstacle and free zone
Obstacles position, Number of white pixels in each segment

Figure 2. Object Detection Process.

4. proposed system

The drone start mission by receive MAVLink message to take off to a specific height (H) and wait for another command from the GCS. We represented the trajectory as many GPS coordinators called waypoints. The proposed system (as explained in Figure 3) consists of three main steps: 1-the guidance stage; 2-the trajectory of the drone; 3-the motion control. We describe each step later in this section.

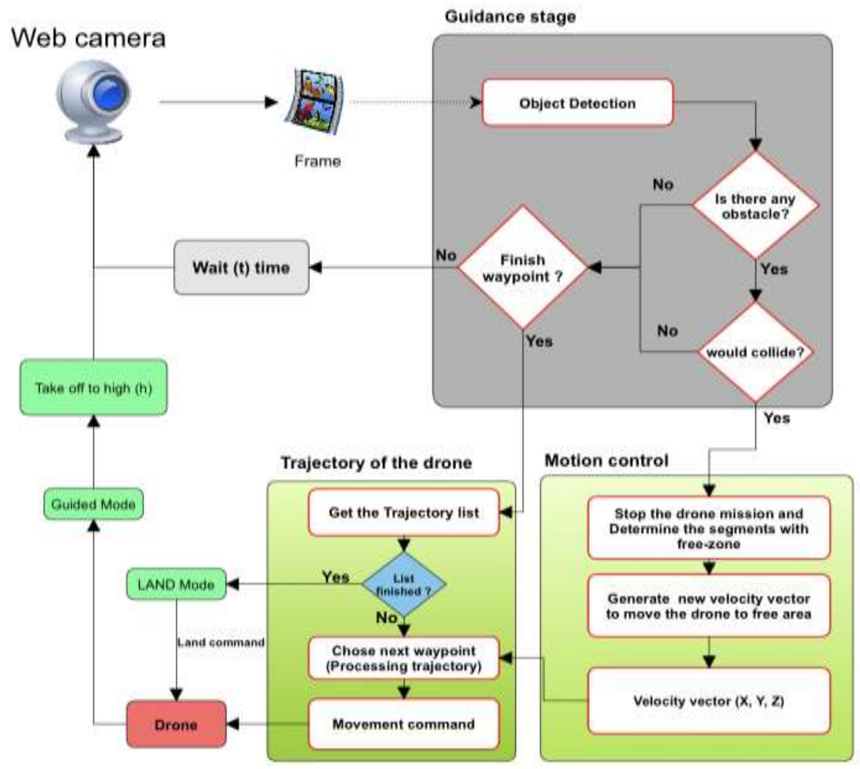

Figure 3. System Overview.

Next, GCS send the first waypoint to the drone to follow its coordinates by using a GPS sensor that uses the location-based movement. After receiving commands, the drone starts moving to the waypoint and listen to GCS that measured the distance between the current location and the destination waypoint. GCS check if the waypoint reaches to execute the next waypoint, we will explain it later in motion control. While the drone executes the current waypoint, images are capture from web camera every (t time) and send to the GCS for processing and detect any object during the mission to avoid them; we will explain this technique next in guidance stage.

We generated a velocity vector depend on the segment of the image that contains obstacle, that makes the drone use velocity based movement instead of using location-based movement. Moreover, the generated velocity vector tells the drone to stop processing the current waypoint and moving to free zone based on the velocity-based movement until avoiding the obstacle.

4.1. Guidance Stage: In this step, we use mathematical treatment on image to detect an obstacle (as explained in Figure 2). The image received from a web camera then uses the following mathematical treatment :

1. Apply a grayscale to the image that captured to reduce the number of colours to the narrower band by using "cvtColor" OpenCV library function that converts it to grayscale.

2. Next, we apply blur to the image by using "filter2D" OpenCV library function by using the following kernel k (shown in Figure 4) that each output pixel is the mean of its kernel neighbours this can reduce the noise from the image.

$$K = \frac{1}{K_{width} \cdot K_{height}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ . & . & . & \dots & 1 \\ . & . & . & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

Figure 4. Kernel that used in "filter2D" OpenCV Library Function.

3. There are different methods used in edge detection, such as(first-order derivative, second-order derivative, and optimal edge detection). In this work, we use an optimal edge detection method called canny edge detection. In OpenCV, we use library function developed by John F. Canny called "Canny" which applies the Canny Algorithm that detecting edges. The results image shows the edge with a low error rate (see Figure 5 .Canny Edge Detection output).
4. to specify the obstacle along with the free zone, we use the following techniques:

   (a) We segment the resulted image into five segments (as showing in Figure 1).
   (b) Count the number of white pixels in each segment individually(the pixels of the grayscale image that are greater than 250) as shown in (Figure 5. Segments).
   (c) Send the calculate result of each segment to the next step (motion control) if obstacles are detected, or send none to (the trajectory of the drone) step if there is no obstacle (Figure 5. Segments_Pixel Count).
   (d) Wait for (t) specific amount of time and repeats the same processes until finishing all the mission.

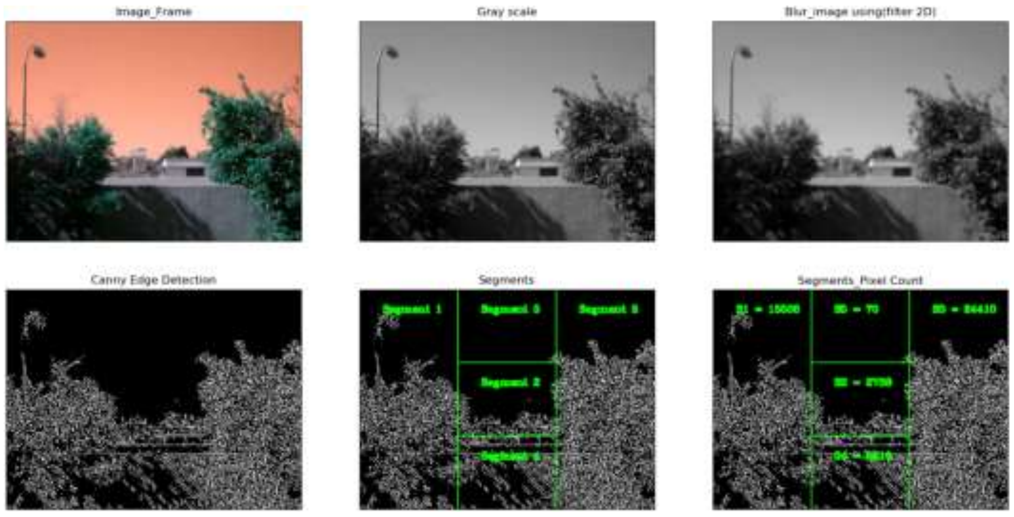

Figure 5. Real Image Capture from Drone Web Camera and Applying the Mathematical Treatment.

4.2. Motion Control: In this step, we deal with generate the movement of the drone for avoiding a collision. Firstly, this step begins when receiving the number of the white pixel for each segment. After that, motion control responds by generated appropriate velocity vector as output to the next step (the trajectory of the drone). The velocity vector commands the drone for move depending on its current position. The avoidance movement showing as (right and left or down and up), this depends on the Y-axis or the Z-axis respectively. The velocity vector generated as showing in the reaction result (Figure 6).

For detecting an object, we check the middle segment (S(2) in reception movement see Figure 1) and check if the segment contains more than thousand of the white pixel (or any optimal value of threshold) that refer to an obstacle in front of the drone. To avoid the obstacle and prevent the drone from a collision, we stop the drone movement immediately and generate the desired velocity vector depend on the free segments with less white pixels calculated. We prefer moving to (the right or the left rather than down or up) for reducing the power needed for avoiding an obstacle.

The resulting velocity vector sent to the next step (the trajectory of the drone) as a value of (X, Y, Z) ready to execute.

```
If  (S(2)  > 1000 pixel)  then

    If  (S(1) < 500 pixel)  and   (S(1) <= S(3))
            Move (left)
    end

    Else if  (S(3) < 500 pixel)  and  (S(3) <= S(1))
            Move (right)
    end

    Else if  (S(4) < 500 pixel)  and  (S(4) <= S(5))  and  (vehicle High > h)
            Move (down)
    end

    Else
            Move (up)

end
```

Figure 6. Reaction Result, S() is the Number of White Pixel in Specific Segment, "h" Represents the Lowest Hight for Fly.

4.3. The trajectory of the drone: In this step, we communicate directly with the drone that can access the low-level control. Our path saved as a sequence of waypoints each has the GPS coordinate as two dimensional (latitude, longitude). While the front of drone has traversable space, we process the waypoint one by one until complete all mission. However, if the drone would make a hit with any obstacles, the drone stop movement and save its current location then start processing the velocity vector (received from motion control step) instead of processing the current waypoint. After moving to the safe area, the drone calculates the distance between (the current location that saved before and the current waypoint), and process the next waypoint if the current waypoint reached.

5. Experimental Results

This experiment was performed using OpenCV 4.1.0 on a raspberry pi4 with 1.5 GHz quad-core processor. We test our method in an outdoor environment with low hight and the system work satisfactorily with avoiding trees, walls, buildings, and even energy transmission lines. The limited computational capacities make our method can deal with one (640*480) image every (t = one second) and leading to test our experiments in flying speed less than 5 kilometres per hour. We can reduce the (t) time between image by resizing the image width and height, but it can affect the detection performance as we tested in real flight. Our experiments focus on the (guidance stage) and generate the proper (motion control) without using the path-planning algorithm for searching the best path in (the trajectory of drone) because of the limited computational capability

6. Conclusions and Future Works

In this work, we have proposed an autonomous navigation system for MAV with close to real-time obstacle avoidance facility. Our method can sense the frontal obstacles and command the hexacopter for moving to a proper direction. The system was tested in the outdoor environment with limited computational capacities and show its ability to avoid frontal obstacle with a minimum amount of fault. The test was showing that the change of the lighting in the environment can affect the performance of the detection as we move to the direction of sunlight. In the future work, we can search for a solution to this issue and combine with the path-planning algorithm so that the MAVs can accomplish some useful task on its own.

References

[1] C. F. Liew, D. DeLatte, N. Takeishi, and T. Yairi, "Recent Developments in Aerial Robotics: A Survey and Prototypes Overview," pp. 1–14, 2017.

[2] M. Iacono and A. Sgorbissa, "Path following and obstacle avoidance for an autonomous UAV using a depth camera," *Rob. Auton. Syst.*, vol. 106, pp. 38–46, 2018.

[3] W. M. Martins, R. G. Braga, A. C. B. Ramos, and F. Mora-Camino, "A Computer Vision Based Algorithm for Obstacle Avoidance," *Adv. Intell. Syst. Comput.*, vol. 738, pp. 569–575, 2018.

[4] H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers, "Collision avoidance for quadrotors with a monocular camera," *Springer Tracts Adv. Robot.*, vol. 109, pp. 195–209, 2016.

[5] J. Byrne, M. Cosgrove, and R. Mehra, "Stereo based obstacle detection for an unmanned air vehicle," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 2830–2835.

[6] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pp. 807–814, 2008.

[7] S. Hrabar, "An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance," *J. F. Robot.*, vol. 29, no. 2, pp. 215–239, 2012.

[8] Y. Yu, W. Tingting, C. Long, and Z. Weiwei, "Stereo vision based obstacle avoidance strategy for quadcopter UAV," *Proc. 30th Chinese Control Decis. Conf. CCDC 2018*, pp. 490–494, 2018.

[9] H. Anis, A. H. I. Fadhillah, S. Darma, and S. Soekirno, "Automatic Quadcopter Control Avoiding Obstacle Using Camera with Integrated Ultrasonic Sensor," in *Journal of Physics: Conference Series*, 2018, vol. 1011, no. 1, p. 12046.

[10] C. Goerzen, Z. Kong, and B. Mettler, *A survey of motion planning algorithms from the perspective of autonomous UAV guidance*, vol. 57, no. 1–4. 2010.

[11] W. G. Aguilar, V. P. Casaliglla, J. L. Pólit, V. Abad, and H. Ruiz, "Obstacle avoidance for flight safety on unmanned aerial vehicles," in *International Work-Conference on Artificial Neural Networks*, 2017, pp. 575–584.

[12] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1750–1757, 2013.

[13] J. O. Lee, K. H. Lee, S. H. Park, S. G. Im, and J. Park, "Obstacle avoidance for small UAVs using monocular vision," *Aircr. Eng. Aerosp. Technol.*, vol. 83, no. 6, pp. 397–406, 2011.

[14] W. Benn and S. Lauria, "Robot navigation control based on monocular images: An image

processing algorithm for obstacle avoidance decisions," *Math. Probl. Eng.*, vol. 2012, pp. 1–15, 2012.
[15] R. Roberts, D.-N. Ta, J. Straub, K. Ok, and F. Dellaert, "Saliency detection and model-based tracking: a two part vision system for small robot navigation in forested environment," in *Unmanned Systems Technology XIV*, 2012, vol. 8387, p. 83870S.
[16] ArduPilot, "ArduPilot Firmware builds." [Online]. Available: https://firmware.ardupilot.org/. [Accessed: 02-May-2020].
[17] MAVLink, "MAVLink Developer Guide." [Online]. Available: http://www.mavlink.org/. [Accessed: 02-May-2020].
[18] "Raspberry Pi." [Online]. Available: https://www.raspberrypi.org. [Accessed: 02-May-2020].
[19] "OpenCV 4.1.0." [Online]. Available: http://opencv.org. [Accessed: 02-May-2020].