Autonomous pilot drones for detecting objects in a space

M Sarosa, M N Zakaria, S Wirayoga[*] and N Muna

Electrical Engineering Department, State Polytechnic of Malang, Indonesia

[*]sepampere@gmail.com

Abstract. The rapid development of technology in the field of air transportation technology has made Drone a widely developed flying vehicle. Drones have many uses for example for hobbyists, navigation tools, and transportation equipment. Drones have also been developed to monitor environments that are difficult for humans to reach. The ability to control manually with limited distance becomes an obstacle. This study describes the Mission Definition System (MDS) which is designed to help in defining the Drone mission easily and efficiently and to support the application of autonomous flight. Under the direct supervision of the pilot line, the selection of Drone control will be possible if in an unsafe situation. After the user defines the mission, it is then forwarded to the Mission Calculation Machine, which is tasked with carrying out the exact calculations to get a compatible and predictable flight plan. A Flight Plan is a list of orders that must be completed by a Drone to completing the designed mission. Test results and measurement of the degree of accuracy of the autonomous system have been tested 5 times for each object and have failed the command once because of natural factors when tested. In terms of the image mapping, drones succeed in detecting objects with only one failure on Saturday by mapping several locations that are targets. For sensors that are used only have one failure to detect obstacles.

1. Introduction

In recent decades, flying technology developers have created variations in technology in accordance with the continued growth of the market and its development for unmanned aerial vehicles (UAVs, Drones), including setting the important manufacturing costs of the technology involved. The infrastructure of inspection, sensing, maintenance, security and agricultural accuracy is only part of the endless aerial platform applications at this time. In this scenario, the complexity of the mission (flight and measurement / data collection specifications) increases dramatically, requiring new techniques and tools to define and carry out flights easily. In this paper, we will focus on the issue of multi-rotor infrastructure inspection, although the techniques and tools involved can be adapted for other applications [1].

The workflow for determining an infrastructure inspection mission involves the user, who might benefit from an aerial view, and a pilot, who actually flies the Drone following the guidelines and requirements imposed by the client. The definition of mission is often done through meetings in the field, where the client explains the purpose of the flight to the pilot, usually with the help of a map, and they approve the plan. From the pilot's point of view, it is quite difficult to follow a flight plan that is not well-defined, examine an area or cover a field, without accurate estimates of flight times, and without real-time feedback on the quality of mission implementation [2].

In this research, we will describe the Mission Definition System (MDS) which is designed to help clients define Drone missions easily and efficiently, and support their application with autonomous flight, under the direct supervision of the pilot line, so that recovery of Drone control will be possible if in unforeseen situations secure. After the user defines the mission, it is then forwarded to the Mission Calculation Machine, which is tasked with carrying out the exact calculations to get a compatible flight plan and predicting compatible trajectories related to the mission. An Flight Plan is a list of orders ordered by the time the Drone has to complete to finish the designed mission (that is, take off, go to the waypoint, then navigate to take a picture, then reach the second waypoint, then navigate again to take a picture, ..., finally landed), while the trajectory was determined through dynamic variations in the time samples of the Drone over time. This trajectory is predicted by considering orders in the Flight Plan, and Drone dynamic behaviour models. After both Flight Plans and trajectories are calculated, they can be represented in the system interface so users can analyse them, check flight duration, and even reproduce drone flight simulations in a 3D simulation environment to check their safety and efficiency.

2. Method

The Mission Definition System (MDS) system consists of several subsystems that work together. The screen-based MDS Hacker Machine Interface (HMI) system (MDS HMI in Figure 1) implements MDS user use cases, utilizing the underlying infrastructure. It can also provide manual access to autopilot drones and control measurements of automatic operation specifications, for cases where the drone system does not have an automatic means to retrieve or receive this data. Meanwhile, the MDS Calculation Engine is the core of the MDS system, which produces complete flight plans and measurement plans from mission specifications, predicts flights according to the expected aircraft dynamics, and stores them in mission repositories that can be accessed via HMI and by other components of the MDS system. The Pilot App System is responsible for providing mission notifications and specifications to the Pilot App, and managing the receipt and termination of these missions by pilots. The final component of the MDS system is the Drone display system, which converts flight plans and measurements in a format suitable for automatic Flight Control systems and measurement control systems in drones. In the simplest approach or explanation, if the drone autopilot allows remote access, it provides direct flight scripts to the drone system at the beginning of the flight [3].

The Pilot App also consists of several components such as having its own HMI which implements use cases. It can also provide manual access to autopilot drones and automatic control measurements according to operating specifications, if the drone is controlled from a cellular application that accepts such specifications. Another component of the Pilot App is the MDS Interface System, which is tasked with receiving mission notifications and specifications from the MDS system, and managing the mission's acceptance by the pilot like Figure 2. Finally, there is the Drone Interface System which changes flight and measurement plans in a format suitable for the autopilot / FCS system and for the measurement control system in the drone. Functionally equivalent to those in the MDS system [4,5].

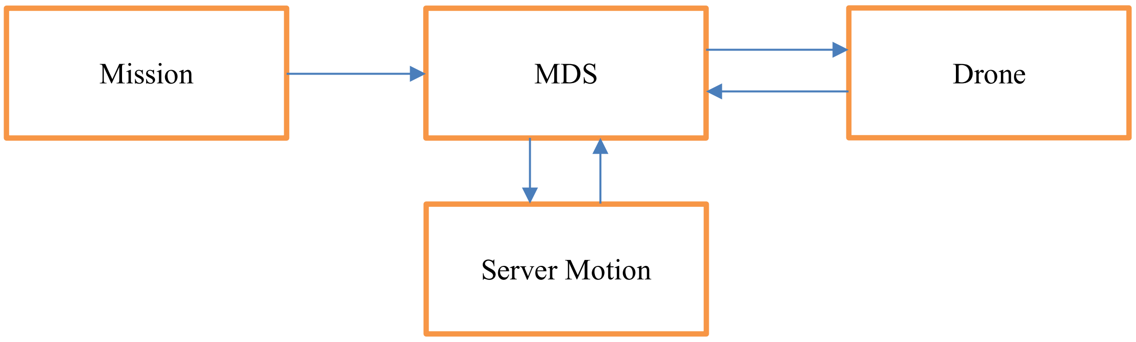


Figure 1. The design of system

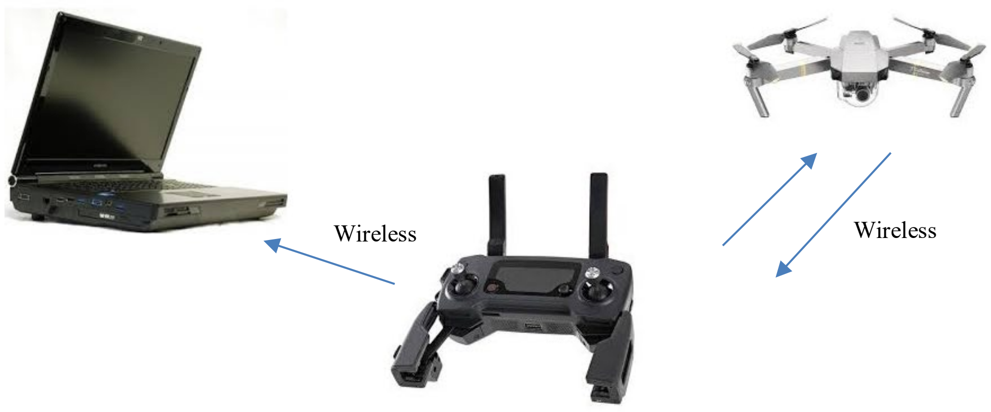


Figure 2. Network design system

Thresholding is a process of changing an image with a degree of gray into a binary or black and white image, so that it can be known which areas are included objects and backgrounds of a clear image[6]. In the thresholding process, it will change the value of an image to a bi-level image that is black (0) for the background (background) and white (255) for the object, or vice versa. A thresholding image will usually be used further for the process of object recognition and feature extraction [7].

The value of a threshold can be determined in several ways, such as: The Iterative Selection method, the Onsu (global thresholding) method, and also by using the Local Thresholding method. In the Iterative Selection method, the threshold value is determined by calculating the average value of the background pixel (Tb) and also from the pixel object (To) in an image. By using the average of the two-pixel values, the threshold value is obtained [8]. The formula used in this calculation is:

$$T = \frac{(\mathrm{To} + \mathrm{Tb})}{2}$$

3. Results and discussion

After conducting the designed method, several results were obtained from two types of experiments. the first experiment was about the ability of the drone's sensors to avoid obstacles in front of it when running automatically [9,10]. And the second result is testing for motion which is used to detect objects. For the results of object detection can be seen in Figure 4 and the results of the detection process are entered into the database server shown in Figure 5. For the first table to be displayed is table 1 about the success of the drone in avoiding obstacles.

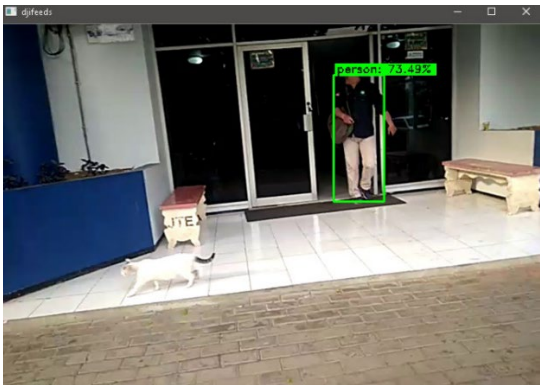
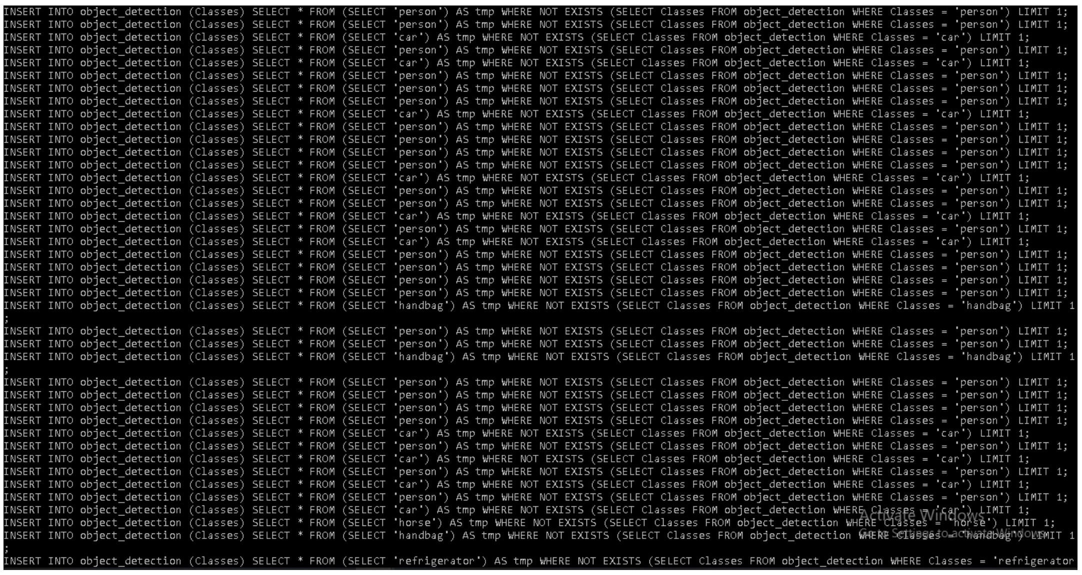


Figure 3. Object detection by drone



```
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'handbag') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'handbag') LIMIT 1
;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'handbag') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'handbag') LIMIT 1
;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'person') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'person') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'car') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'car') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'horse') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'horse') LIMIT 1;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'handbag') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'handbag') LIMIT 1
;
INSERT INTO object_detection (Classes) SELECT * FROM (SELECT 'refrigerator') AS tmp WHERE NOT EXISTS (SELECT Classes FROM object_detection WHERE Classes = 'refrigerator
```

Figure 4. The process from detection to database

Table 1. The results avoid obstacle by sensor.

No	Object	Sensor Ability Test				
		1	**2**	**3**	**4**	**5**
1	Tree	Success	Success	Success	Success	Success
2	Wall	Success	Success	Success	Success	Success
3	Mirror/Glass	Success	Success	Fail	Success	Success
4	Big Stone	Success	Success	Success	Success	Success
5	Human	Success	Success	Success	Success	Success

From Table 1 it can be seen that from 5 experiments each different object. It is known that when getting a barrier to trees, large stones, walls, and humans, the drone managed to avoid it 5 times as well or the success rate reached 100 percent. As for objects in the form of glass, drones fail once in the process of avoiding obstacles. This is due to the nature of the glass that is less suitable in ultrasonic reflection on the sensor used by the drone. Then let see Table 2 for Motion Detection.

Table 2. The results of motion detection by range.

No	Object	Range of Motion Detected (Meter)				
		2	**3**	**4**	**5**	**6**
1	Human	Success	Success	Success	Success	Fail
2	Car	Success	Success	Success	Success	Success
3	Motorcycle	Success	Success	Success	Success	Fail
4	Animal 4 Legs	Success	Success	Success	Fail	Fail
5	Bus	Success	Success	Success	Success	Success

From Table 2 it can be seen when the distance is up to 6 meters, only large objects that do not experience failures are buses and cars. As for objects in the form of motorbikes, quadrupeds, and humans fail when the distance is increased. Where motorbikes and humans experienced 1 failure while four-legged animals failed 2 times. This is because the training data created is not optimal for long distances and requires more samples for training data.

4. Conclusion
In this research, A Drone can fly properly if looking on the ability sensor from it. A drone can detection 5 obstacle object with many times of success. When a drone detection Tree, Wall, Big Stone, and Human, it gets 5 success for detection them. When a drone detection Mirror/Glass, it gets 1 fail and 4 success in trial sensor of automated drone. Detection programs that are created can integrate well with systems owned by automated drones. the program can compensate for the speed of drones in conducting surveys in areas captured by drone cameras. of the many objects attempted, only often failing at a distance of 6 meters while below that dominated successful detection.

5. Acknowledgment
This research was funded by Malang State Polytechnic DIPA Fund. Thank you for that.

References
[1] Bjarne K 2016 Autonomous Drone with Object Pickup Capabilities. Norwegian University of Science and Technology.
[2] Luke K and James H 2017 Development of Autonomous Quadcopter *IEEE*.
[3] Muhammad F Romdhoni, Johannes Adiyanto, dan Hendi Warlika Sedoputro. 2016. Penggunaan Drone sebagai Media Digitasi Penggambaran 3 Dimensi Bangunan dan Pemetaan Kawasan. TEMU ILMIAH IPLBI 2016.
[4] Juan A. Besada, Luca Bergesio, Iván Campaña, Diego Vaquero-Melchor, Jaime López-Araquistain, Ana M. Bernardos, dan José R. Casar. 2018. Drone Mission Definition and Implementation for Automated Infrastructure Inspection Using Airborne Sensors. Sensors 2018, 18, 1170; doi:10.3390/s18041170.
[5] Ludovic Apvrille, Tullio Tanzi, dan Jean-Luc Dugelay. 2014. Autonomous Drones for Assisting Rescue Services within the context of Natural Disasters. https://www.researchgate.net/publication/271834672.
[6] W. M. Hu, T. N. Tan, L. Wang and S. Maybank, "A survey on visual surveillance of object motion and DOI: 10.18535/ijecs/v5i5.11 Shilpa, IJECS Volume 05 Issue 5 May 2016 Page No.16376-16382 Page 16382 behaviour", IEEE Transactions on Systems, Man and Cybernetics, vol.34, no.3, pp. 334-352, Dec 2004.
[7] J.Joshan Athanesious, P.Suresh, "Systematic Survey on Object Tracking Methods in Video", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) pp. 242-247, October 2012.
[8] Kinjal A Joshi, Darshak G. Thakore, "A Survey on Moving Object Detection and Tracking in Video Surveillance System", International Journal of Soft Computing and Engineering, Vol. 2, Issue-3, July 2012.
[9] Sunitha M.R, H.N Jayanna, Ramegowda, "Tracking Multiple Moving Object based on combined Color and Centroid feature in Video Sequence", in IEEE International Conference on Computational Intelligence and Computing Research, pp. 846-850, Dec 2014.
[10] Gandham Sindhuja, Dr Renuka Devi S.M, "A Survey on Detection and Tracking of Objects in Video Sequence", International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015.