NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ASSESSMENT OF FOREIGN OBJECT DEBRIS MANAGEMENT USING GROUP 1 UNMANNED AERIAL SYSTEMS

by

Wee Leong Lee

September 2018

Thesis Advisor: Oleg A. Yakimenko
Second Reader: Fotis A. Papoulias

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ASSESSMENT OF FOREIGN OBJECT DEBRIS MANAGEMENT USING GROUP 1 UNMANNED AERIAL SYSTEMS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Wee Leong Lee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The level of unmanned aerial system (UAS) technology provides an opportunity to improve productivity and at the same time increase safety in airport operations. This thesis aims to study the requirements of a system of Group 1 UAS swarm to perform foreign object debris (FOD) management at an airport. A concept of operations of the Automated FOD Detection System (AFDS) was designed, and the graphical user interface (GUI) was developed to prove the concept. A variety of image processing algorithms was developed in MATLAB to perform the functions of AFDS. FOD parameters such as its color and size, as well as the operating altitude of UAS and the image processing filter window size, were varied to determine the optimal AFDS configuration. This research involved hands-on experience of assessing the developed algorithms on board a commercial-off-the-shelf DJI UAS in a series of experiments conducted over a real runway at Camp Roberts, CA. The performance of the system in a large airport was assessed and the costs involved in implementing the system were determined. The results showed that FOD management using Group 1 UAS is feasible, with future work required in developing control algorithms of the UAS and testing them to eventually operationalize the AFDS.

14. SUBJECT TERMS runway safety, foreign object debris, unmanned aerial system, image processing	**15. NUMBER OF PAGES** 89
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ASSESSMENT OF FOREIGN OBJECT DEBRIS MANAGEMENT USING GROUP 1 UNMANNED AERIAL SYSTEMS

Wee Leong Lee
Major, Singapore Air Force
Bachelor in Engineering, National University of Singapore, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2018

Approved by: Oleg A. Yakimenko
Advisor

Fotis A. Papoulias
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The level of unmanned aerial system (UAS) technology provides an opportunity to improve productivity and at the same time increase safety in airport operations. This thesis aims to study the requirements of a system of Group 1 UAS swarm to perform foreign object debris (FOD) management at an airport. A concept of operations of the Automated FOD Detection System (AFDS) was designed, and the graphical user interface (GUI) was developed to prove the concept. A variety of image processing algorithms was developed in MATLAB to perform the functions of AFDS. FOD parameters such as its color and size, as well as the operating altitude of UAS and the image processing filter window size, were varied to determine the optimal AFDS configuration. This research involved hands-on experience of assessing the developed algorithms on board a commercial-off-the-shelf DJI UAS in a series of experiments conducted over a real runway at Camp Roberts, CA. The performance of the system in a large airport was assessed and the costs involved in implementing the system were determined. The results showed that FOD management using Group 1 UAS is feasible, with future work required in developing control algorithms of the UAS and testing them to eventually operationalize the AFDS.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AOA	Air Operations Area
ATC	Air Traffic Control
AFDS	Automated FOD Detection System
CMOS	Complementary Metal-Oxide-Semiconductor
FAA	Federal Aviation Authority
FCC	Federal Communications Commission
FOD	Foreign Object Debris
FOV	Field of View
GSD	Ground Sample Distance
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
NPS	Naval Postgraduate School
IFOV	Instantaneous Field of View
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Limitations in manpower in Singapore have led to the desire to improve productivity and efficiency in operations throughout the country, including airport operations. At the same time, foreign object debris (FOD) is a significant concern in airport operations. Together with recent trends of improvements in unmanned aerial systems (UAS) technology and clarification of regulations regarding small UAS operations, it is an opportune time to study the feasibility of performing FOD management using small (Group 1) UAS.

The concept of operations for the Automated FOD Detection System (AFDS) involve a UAS swarm lined up at the start of the runway, which on command from the operator, would proceed along the runway to take images at short intervals, covering the length of the platform where the inspection was to occur. While traversing the runway, the images taken would be transmitted to the central processor to be processed. The process would be controlled by the graphical user interface (GUI) installed on a desktop computer in the control tower. The operator would be informed of any FOD detected through the GUI.

The image processing entails the following steps:

1. Read background image file which forms the baseline.
2. Read current image file where FOD may be detected.
3. Convert and filter the images into grayscale.
4. Align the two images.
5. Find the difference of the current image from the background image.
6. Filter the results to sharpen the resultant binary image.
7. Draw bounding boxes around image objects which are suspected FODs.
8. Overlay the bounding boxes on the current image file.

Experiments were conducted at Camp Roberts to determine the optimal configuration of the AFDS that is able to detect objects as small as 3cm-by-3cm. Object parameters that were varied in the experiments were as follows:

1. Object color
2. Object size
3. UAS altitude
4. Image processing filter window size

The results of the experiments found that white specimens were detected more reliably than black ones, with white objects being able to be detected by the computer algorithm 91.3% of the time, compared to 35.8% for black objects. The detection rate increased as object size increased. For black objects, the smallest 3cm-by-3cm object was detected 28.3% of the time, compared to 40.8% of the time for the largest 20cm-by-20cm object. For white objects, the smallest object was detected 69.2% of the time, while objects that were 10 cm or larger were detected 100% of the time.

As the UAS altitude increased, the detection rate decreased. Black objects were detected 96.7% of the time at 1 m UAS altitude, which decreased to 10% at 10 m. White objects were detected 100% of the time at 1 m UAS altitude, which decreased to 47% at 10 m.

As the filter window size increased, the detection rate decreased. Black objects were detected 98% of the time with the smallest 3pix-by-3pix median filter, which decreased to 10% with the largest 25pix-by-25pix median filter. White objects were detected 100% of the time with the 3pix-by-3pix median filter, compared to 78% with the 25pix-by-25pix median filter.

With the results from the experiments, the optimal AFDS configuration was selected based on the ability to detect black objects. From Table ES-1, the AFDS configuration that was selected was the UAS operating at 4 m with a filter window size of 7pix-by-7pix.

Table ES-1. Detection rate of black objects in the UAS altitude-filter window size coordinate

Filter window size (pix) \ UAS altitude AGL (m)	1	2	3	4	5	6	7	8	9	10
3	100%	100%	100%	100%	100%	100%	100%	100%	100%	80%
5	100%	100%	100%	100%	100%	100%	100%	80%	40%	40%
7	100%	100%	100%	100%	60%	80%	20%	40%	0%	0%
9	100%	100%	100%	60%	20%	0%	20%	0%	0%	0%
11	100%	100%	80%	40%	0%	0%	0%	0%	0%	0%
13	100%	80%	60%	40%	0%	0%	0%	0%	0%	0%
15	100%	80%	0%	0%	0%	0%	0%	0%	0%	0%
17	100%	80%	0%	0%	0%	0%	0%	0%	0%	0%
19	100%	60%	0%	0%	0%	0%	0%	0%	0%	0%
21	100%	40%	0%	0%	0%	0%	0%	0%	0%	0%
23	80%	20%	0%	0%	0%	0%	0%	0%	0%	0%
25	80%	20%	0%	0%	0%	0%	0%	0%	0%	0%

With the AFDS configuration determined, the amount of resources required to perform a runway inspection was calculated. For a large airport with a 4000m-by-60m runway such as the Singapore Changi Airport, 11 UAS are required to cover the width, and each UAS would be required to take 964 images to cover the length of the runway. An airport with a smaller runway of 2187m-by-45.72m at the Monterey Regional Airport would require 9 UAS with 527 images each. The 4000m-by-60m runway would take 13.3 minutes to be scanned, and almost 100 hours for the images to be processed on a laptop computer, which could be reduced to less than an hour with compiled code. While that is still too slow to be practical in the operational environment, code optimization and better hardware should result in improvements to make the AFDS a reality in the near future.

Finally, the cost of the AFDS was presented. For a large airport such as the Singapore Changi Airport, it would cost approximately $55,500 to implement the system, assuming a parallel computing system with one computer processing the inputs from each individual UAS.

The thesis presented the requirements of a system of Group 1 UAS swarm to perform FOD management at an airport, and found it feasible. Future work in UAS control and testing would improve the system and allow it to eventually be operationalized.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Working away from home and loved ones had not been an easy process. I would like to thank my wife, Shu Wei, for having the patience and tolerating me being away from her physically and at times even electronically during the past year. I am coming back soon!

I would like to thank Professor Yakimenko for the tremendous guidance he gave me during this thesis process. My many thinking, coding and writer's blocks quickly became unstuck after the quick discussions we had at your office. Thank you also for the opportunities to share my knowledge with a wider audience. I hope my applications could be of use to both the Singapore Navy and U.S. Navy in the future!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter describes the background and the impetus for the thesis, followed by the problem statement that the thesis aims to address.

A. NEED FOR AUTOMATION IN SINGAPORE

Singapore has limitations in manpower; the annual Total Fertility Rate in Singapore had been declining since the 1970s, with the latest figures at 1.16 for 2017 (Ong 2018). This meant that the population was not reproducing sufficiently to replace itself. The 2013 Population White Paper stated that without immigration, the citizen population would shrink from 2025. However, infrastructure concerns had led to the government restricting the intake of new immigrants (Seow 2017). These developments have led to calls from the government to both the public and private sectors to seek ways to improve productivity and efficiency, and reduce the reliance on manpower (Pedersen 2017).

B. AIRPORT OPERATIONS

Airport operations are an example where manpower needs are high. Airport operations could be broadly separated into landside and airside operations. In page three of his course notes on "Introduction to Airports Design and Operations," Dr. Sherry (2009) described landside as "how passengers arrive/depart the airport terminal building and move through the terminal building to board the planes," and airside as "movement of the airplanes on the airports surface." To the layman, landside operations are those that are that the general public could interact with first-hand, such as ticketing, check-in, food and retail among others. Airside operations are typically those that are performed away from the general public, such as aircraft turnaround checks and maintenance, Air Traffic Control (ATC), and airfield maintenance.

1. Critical Task of Detecting and Removing Foreign Object Debris

Foreign object debris (FOD) is a significant concern in airport operations. McCreary's study in 2010 reported that the aviation industry estimated the worldwide cost of FOD to be between US$2.3 billion to US$4 billion annually. In his bottom-up study, he

found that the direct costs of on-runway FOD were even higher, on the order of US$7 billion a year. More significantly, FOD had been attributed as one of the significant, if not the direct cause of several aviation crashes, the most high-profile of which was the Air France Concorde crash in July 2000 that killed 109 on-board and four on the ground (Alexander-Adams 2013). At the same time, FOD detection by (human) visual means is not reliable. In the same study, McCreary (2010) compared the performance of visual inspection to automated scanning and found that with visual inspection, one piece of FOD would be found on the runway every two months, as compared to one found every two days with automated scanning.

2. UAS as a Possible Solution

One of the ways to tap on the potential of automation, while reducing the reliance on manpower is to employ robotics (Chambers and Rohaidi 2017) such as mini-Unmanned Aerial Vehicles (UAV) or Systems (UAS). UAS technology had improved, while costs have reduced in the past year for relatively high-end consumer-grade UAS to be affordable to the public (Wile 2017). The reduction in costs of such UAS presents an opportunity for the military to tap on its potential, and reduce the need for manpower in areas and operations that these systems are deployed in.

C. VARIETY OF UAS AND TRENDS

UAS could be classified under two major categories. They are either fixed-wing, or rotorcraft. UAS have generally the same physical characteristics as the manned aircraft of the same types, and largely the same flight characteristics. Table 1 shows the characteristics of fixed-wing and rotorcraft UASs.

Table 1. Characteristics of different UAS types.
Adapted from Hogan et al. (2017).

	Fixed-Wing	Rotorcraft
Characteristics	• Large payload capacity • High top speed • Long flight time • Long range	• Highly maneuverable • Ability to hover, rotate and capture images at almost any angle • Easy learning curve • Many models with range of costs

1. Regulatory Trends

The areas where UASs could be deployed in airport operations are numerous. Hubbard et al. (2017) proposed employing UAS in performing obstruction analysis, pavement condition assessment and inspection, airfield light inspections, wildlife management, security, emergency response, and constructions. Many of these operations have already been performed by UAS outside of airfields, but with the publication of the new Small UAS Rules (Part 107) by the Federal Aviation Administration (FAA), which cover Groups 1 (up to 20 lb) and 2 (21 - 55 lb) UAS, there now is a framework to operate these systems legally in the United States. Such a development provides Singapore with the opportunity to implement UAS for our many operations, while keeping operational risks to a minimum.

2. UAS Technology Application Trends

This nascent technology in UAS had in fact been trialed at various airports in the recent past. After being reported of using UAS to perform surveys of a parking garage at the Hartsfield-Jackson International Airport in Atlanta in January 2017 (Worland 2017), the airport further explored the opportunities by conducting subsequent trials of having UAS perform inspection of the runway for damage (Whitehead 2017). UAS had also been reported to be integrated into the airport operations at the Edmonton International Airport in Alberta, Canada, with the airport working with private companies Aerium Analytics and Clear Flight Solutions to employ UAS for wildlife management (PR Newswire 2017). While these activities show that the potential of commercial prospects of UAS in airport operations, the underlying technology have yet to be understood fully.

3. FOD Detection with Automated Means

Performing FOD detection with UASs involves automated image (or scene) processing. At least two companies had published reports documenting their approaches in this field, with the Israeli company Pharovision deploying the Sentinel system (with video charge-coupled device image and infrared sensors) to perform FOD detection in 2014 at an Israel Defence Force Air Base. The report stated that the system was able to detect objects of 3 cm x 3 cm from a distance of 800 meters (Leite 2014). The Singaporean

company Stratech had even patented a surveillance system for detecting FOD that involved cameras to capture runway images, and adaptive image processing (using global histogram and statistical analysis of preceding images) of the images captured by the cameras (Chew 2011). This technology had been implemented at the Singapore Changi Airport since 2008.

D. PROBLEM FORMULATION

This thesis aims to study the requirements of a system of Group 1 UAS swarm to perform the FOD management at an airport. The UAS swarm is envisioned to collect images of the Air Operations Area (AOA). These images would then be aligned to those of a "clean" image to optimize system's performance to detect the presence of FOD. Parameters relating to the size and color of the FOD, the altitude of the UAS, as well as the image processing filter size would be varied to determine a recommended configuration for the system.

The thesis is organized as follows. Chapter I reviews the need for UAS and its application in airport operations. Chapter II examines the operational considerations for using the UAS swarm to perform FOD detection and the graphical user interface (GUI) mock-up of the system. Chapter III illustrates the challenges in image processing. Chapter IV discusses the algorithm development and data gathering process. Chapter V presents the data analysis, and finally in Chapter 6 the conclusions and recommendations are made.

II. OPERATIONAL CONSIDERATIONS AND GUI MOCKUP

The operational considerations for performing airfield inspections are first examined in this chapter. We would then present the concept of operations for the Automated FOD Detection System (AFDS) that involves the use of a UAS swarm to perform FOD inspection. We would also present in this chapter the mockup for the GUI for the system.

A. AIRFIELD INSPECTION OPERATIONAL CONSIDERATIONS

The Airfield Operations Procedures and Programs (Department of the Air Force 2015) describes the requirements for performing an airfield inspection. Paragraph 17.1.6.6 in particular describes the inspection of paved areas, for conditions that would cause concern for aircraft operations. These conditions include cracks, holes, rubber deposits, vegetation growth, and FOD and contaminants.

Paragraph 17.1.7 of the same document discussed the inspection techniques, which entailed airfield inspectors driving on the runway towards the direction of landing aircraft for safety of the personnel. The document also suggested having a varied inspection pattern would reduce complacency and missing out on items that otherwise would need to be corrected.

The FAA's Advisory Circular 150/5220-24 dated September 30, 2009, on Airport FOD Detection Equipment described the basic functions and their performance requirements for FOD detection equipment. The FOD detection equipment would need to be able to detect single and multiple FOD items, and provide an alert to the user when FOD was detected. Among the objects that the system must detect were an unpainted metal cylinder of 3.1 cm (1.2 in) high and 3.8 cm (1.5 in) in diameter, and a white, grey or black sphere of 4.3 cm (1.7 in) diameter which is the size of a standard golf ball.

B. CONCEPT OF OPERATIONS

The concept of operations of the AFDS involves a UAS swarm lined up at one end of the runway (or platform to be inspected). The number of UASs required differs

according to the width of the runway and the altitude of the UAS. On command from the operator through the GUI, the UAS would proceed to take images along the runway at short intervals, such that images of the entire runway would be taken. Spot inspection of specific areas could also be selected through the GUI. The GUI would be installed on a desktop computer located in the control tower, although alternative locations such as the base operations center could be equipped with the software as well. Figure 1 shows a simulated swarm of UAS performing inspection of a runway.

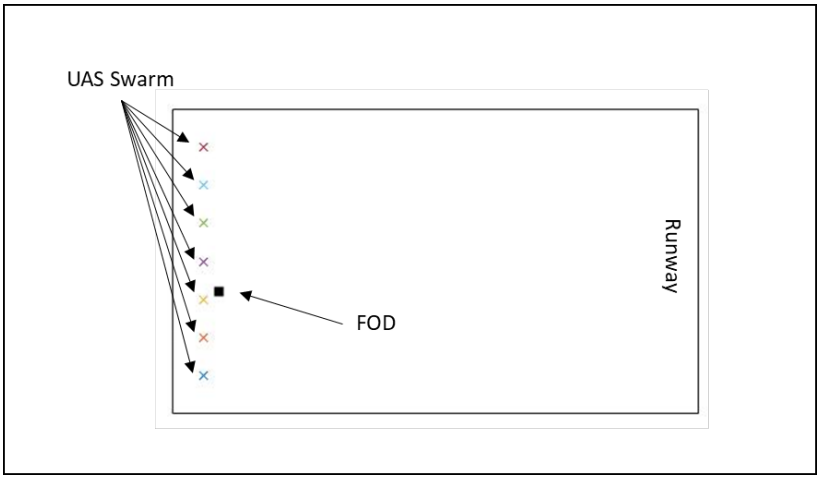


Figure 1. UAS swarm performing inspection on runway

While traversing along the runway, the UASs would transmit the images that had already been taken to the central processor to process the images. The images taken during the inspection would be processed against the baseline images that were taken beforehand, which were FOD-free.

If any FOD were detected, the operator would be informed through the GUI with the image of the FOD and the location. Figure 2 shows the high-level graphical description of the concept of operations.

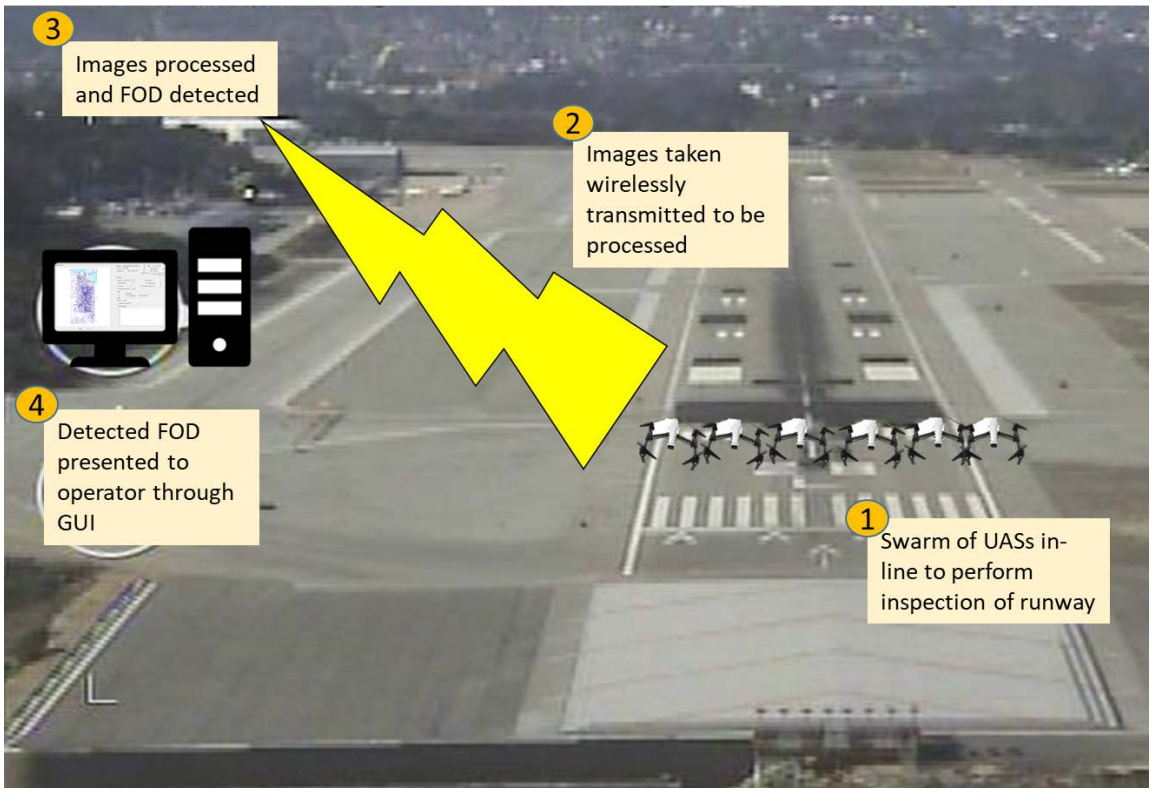


Figure 2. Graphical description of the concept of operations for the AFDS.

C. FUNCTIONAL ANALYSIS OF THE AFDS

A functional analysis was performed to determine the critical functions of the AFDS, and to organize them into a hierarchical form. The critical functions of the AFDS were determined to be 1) provide platform for the sensor, 2) scan area of interest, 3) process output from sensor, and 4) interface with operator. These critical functions were further decomposed into sub-functions, as shown in Figure 3.

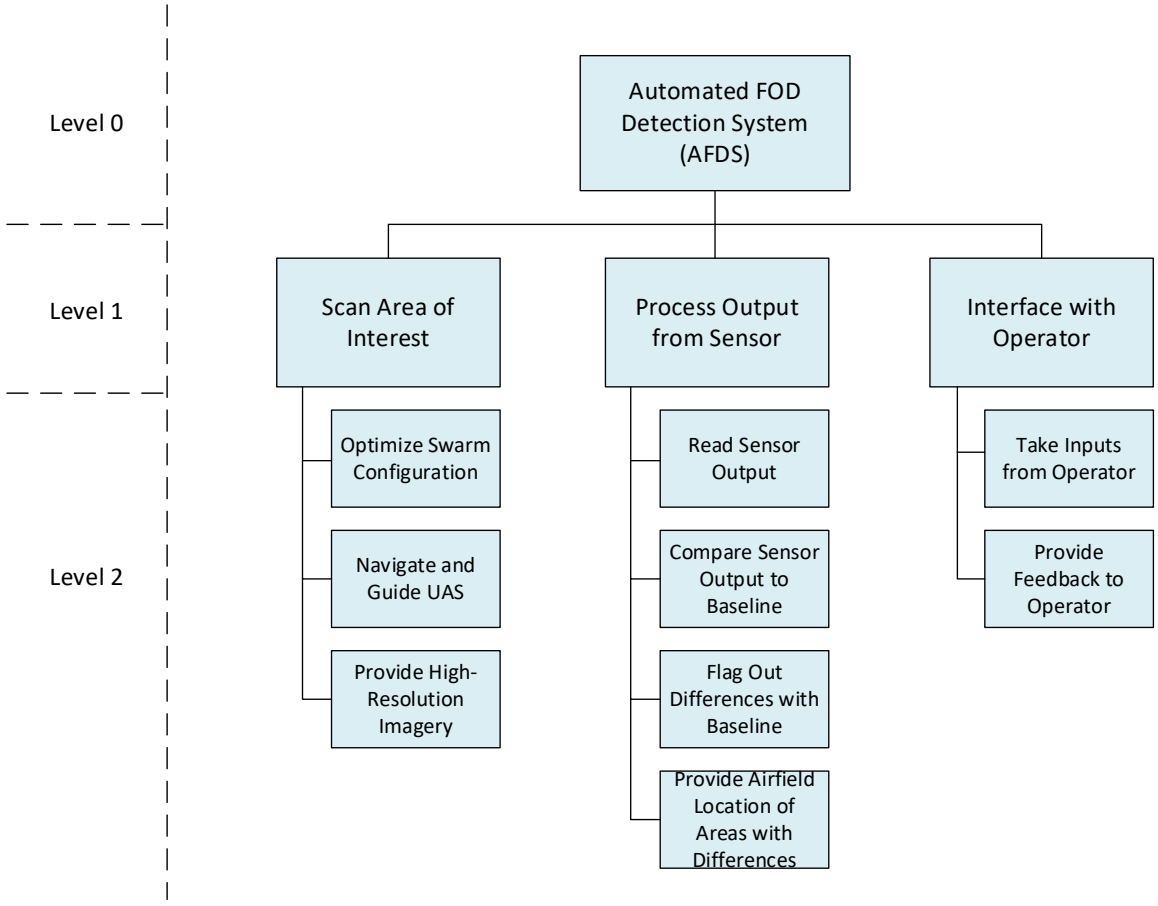


Figure 3. Critical functions of AFDS in a functional hierarchy

1. Scan Area of Interest

The system must scan the area of interest, by optimizing the UAS swarm and then guiding the UAS to navigate over the area. It must also provide high-resolution imagery of the area of interest.

2. Process Output from Sensor

The images transmitted are first read into the processor and compared to the baseline images present in the database. Differences from the baseline are flagged out as potential FOD, and the location where these differences are found are marked.

3. Interface with Operator

The potential FOD and their locations would be presented to the operator through an interface. The operator could also command the UAS swarm to perform inspection of specific locations in the airfield through the interface.

D. GRAPHICAL USER INTERFACE MOCKUP

The GUI for the AFDS is divided into four different areas, shown in Figure 4.

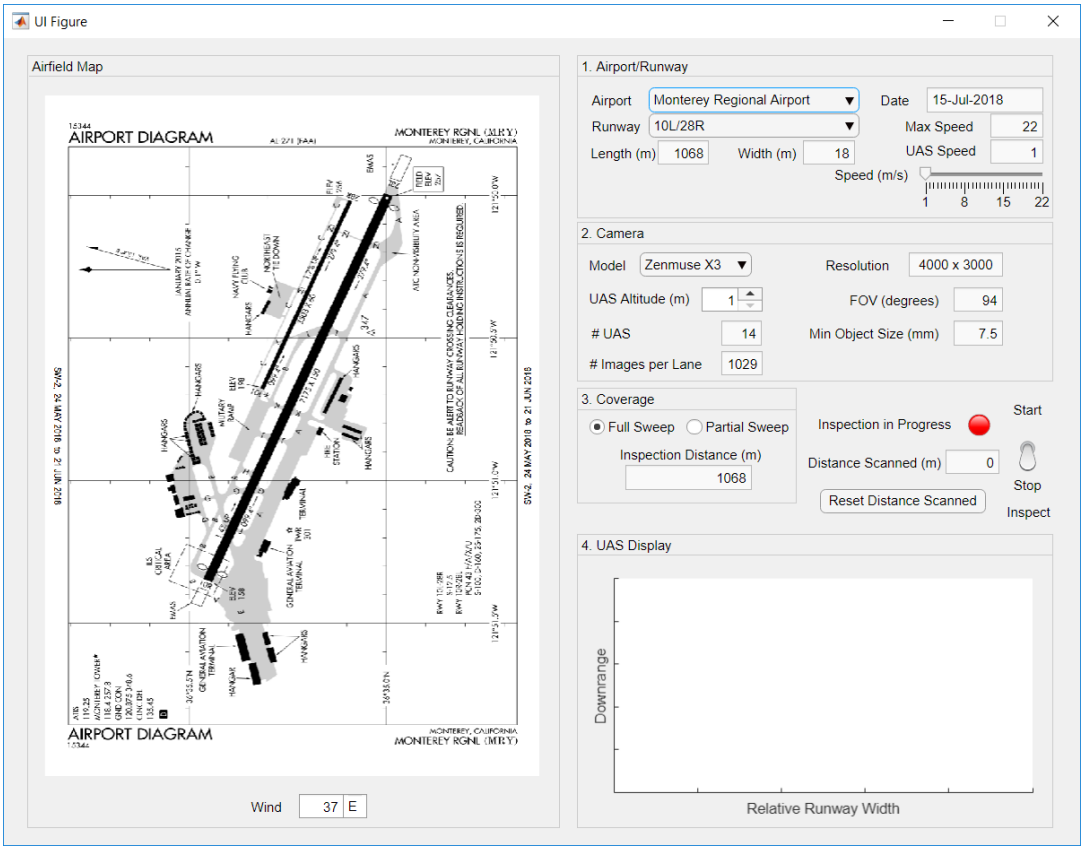


Figure 4. GUI mock-up of AFDS

In the first area in the top right corner of the application window, the operator would select the airport and the runway of interest. The runway length and width would then be updated in the GUI. The user could then select the speed at which the UAS swarm would traverse along the runway.

In the second area, the operator could select the camera model to perform the airfield inspection. Selecting different camera models provide the operator with the ability to change the minimum object size that the AFDS could detect. In the process of selecting a different camera model with a different field of view (FOV), the number of UAS required to perform the inspection would differ, together with the number of images required to be taken to cover the whole area.

In the third area, the operator could select the coverage of the inspection, which are "full sweep" which indicates a full inspection of the runway, or "partial sweep," which indicates the inspection of a selected area. If "partial sweep" is selected, a new window will pop-up with crosshairs which allows the operator to select the start and end points of the inspection on the airfield map. See Figure 5.

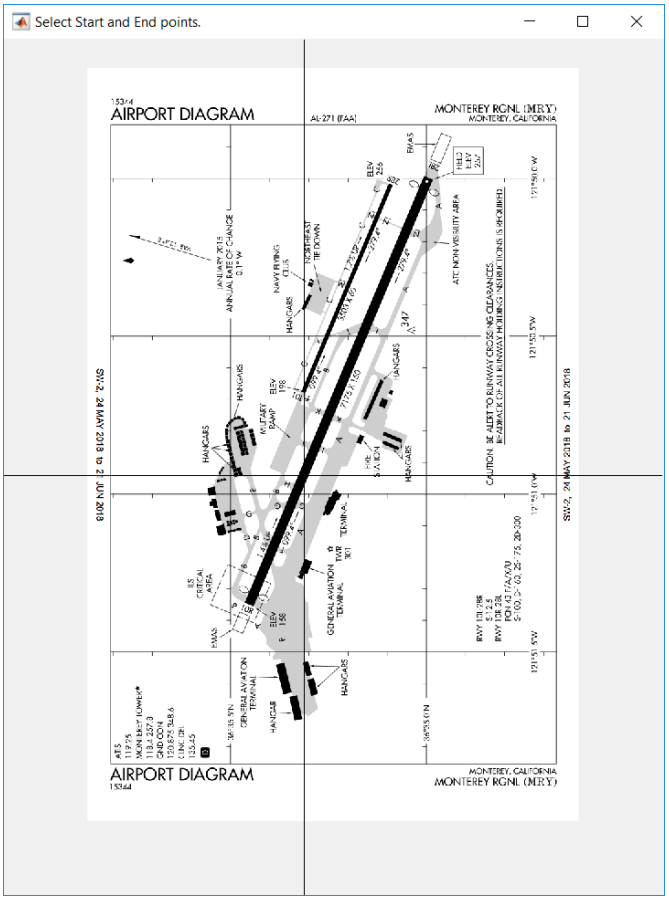


Figure 5. Pop-up window for selection of start and end points of inspection

Once the area to be scanned is selected, the Inspection Distance field would show the length of the runway or taxiway that the inspection would take place. The operator can then use the toggle switch to start the inspection. When the inspection is ongoing, the indicator light for "Inspection in Progress" would turn green, while the "Distance Scanned" field would show the progress of the UAS in meters, as shown in Figure 6. At any time during the inspection, the operator can stop and resume the process by selecting the toggle switch. If the operator requires the inspection to be restarted, he would click on the "Reset Distance Scanned" button to set the distance scanned back to zero and close the windows showing the FODs found during this iteration.

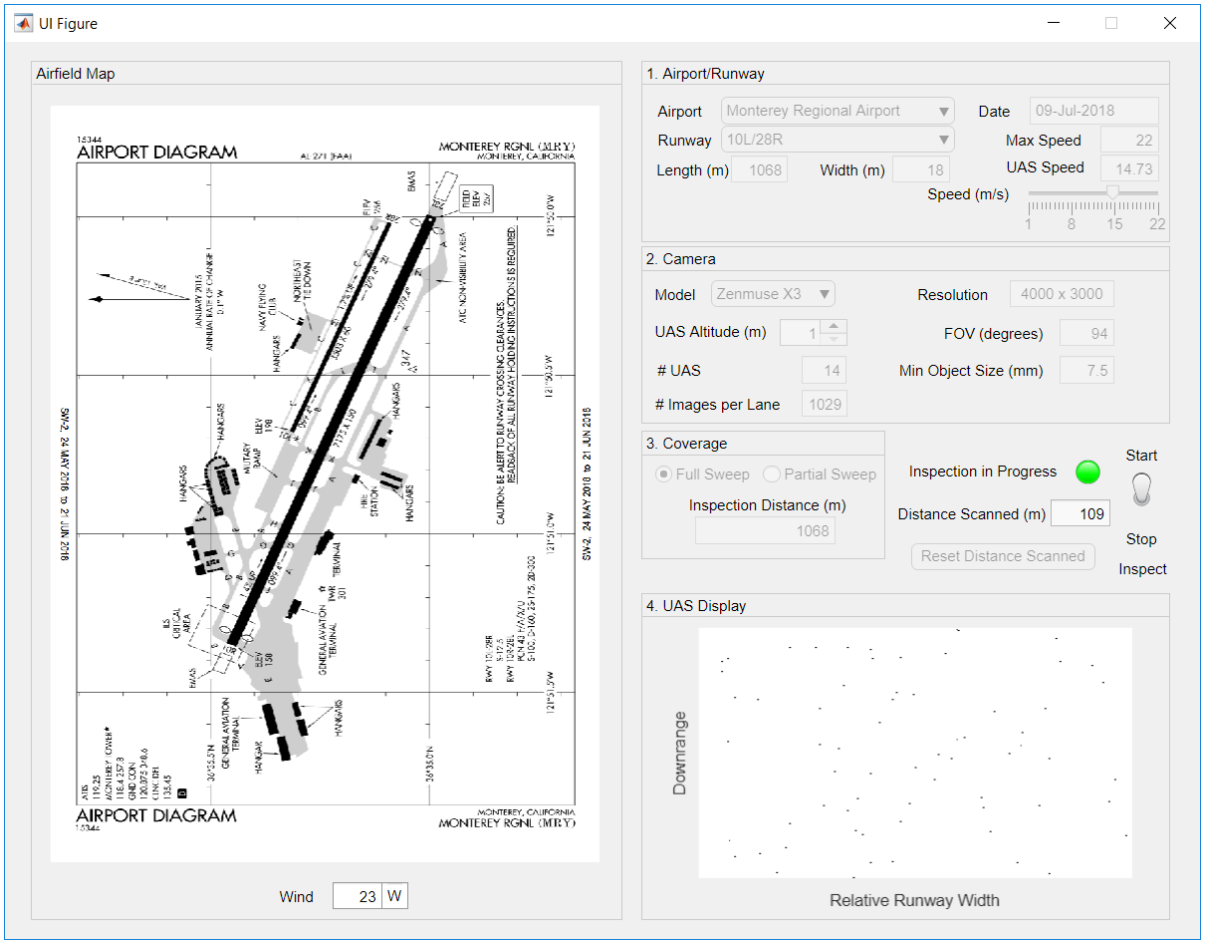


Figure 6. Ongoing inspection indicated by the green light beside the "Start/Stop" toggle switch

Finally, in the fourth area, the UAS display shows the real-time images collected by the UAS swarm. Should any FOD be detected, a new window would pop-up, with the location of the FOD in distance from origin of the inspection sequence depicted in the title bar. For example, Figure 7 shows the image of an FOD (the black square of cardboard) that was detected at 85 m from the start of the inspection sequence.

Refer to Appendix A for the MATLAB code for the GUI.

Figure 7. Sample window showing image of FOD detected by the UAS

III. CHALLENGES IN IMAGE PROCESSING

A. IMAGE PROCESSING DEFINITIONS

To be able to detect FOD in the images collected by the UAS, those images need to be processed by computers. Harney (2004) defines an **image** as "a distribution in at least two spatial dimensions of one or more parameters related to physical properties of an object or scene." In other words, the physical properties of the world are represented by two or more of the spatial dimensions which are listed in Table 2.

Table 2. Spatial dimensions and physical properties. Source: Harney (2004).

Potential Spatial Dimensions	**Potential Physical Properties**
Azimuth (or Bearing)	Color
Elevation Angle	Reflectivity
Range	Reflected Intensity
Cartesian Coordinates (x, y, z)	Radiance
Depth (or Altitude)	Concentration
Map Coordinates	Transmittance (or Absorptance)
Cross-Range	Velocity
	Temperature
	Range
	Radar Cross Section

Hence, in this thesis, one could consider an image as a set of physical properties such as color and reflected intensity that are represented by the spatial dimensions of Cartesian coordinates and altitude.

Harney (2004) uses the term **pixel** (for "picture element") to "denote any one of the discretely-addressable regions in the image space from which an image is constructed by assignment of parameter values," while the term **resel** (for "resolution element") is used to describe "any region of image space whose size in each dimension is equal to the system resolution in that dimension."

Harney defines the **field of view (FOV)** of a system as the range of angles that is scanned and/or subtended by the detector array. He defined the **instantaneous field of view (IFOV)** as the angular portion of the FOV that can be seen by one detector at any instant in time.

The **ground sample distance (GSD)** is defined as the distance between the centers of two consecutive pixels on the ground. The parameters that affect the GSD are the sensor width S_W (in mm), the focal length F_R(in mm), the distance from the image, or flight height H (in m), and the image width D_W (in m). See Figure 8.

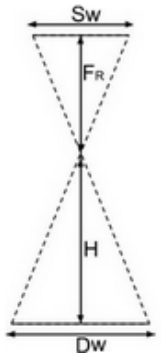


Figure 8. Parameters affecting GSD. Source: PIX4D (2017).

For an image with 4:3 ratio, the focal length F_R is given by

$$F_R[mm] = \frac{F_{35} \times S_W}{34.6} \tag{1}$$

where F_{35} is the focal length that corresponds to the 35 mm equivalent.

From trigonometry, the ground distance covered by the UAS at 4 m is given by the equation

$$\frac{H}{F_R} = \frac{D_W}{S_W} \tag{2}$$

and GSD is given by the equation

$$GSD = \frac{D_W}{\text{Image Width [pixel]}} \tag{3}$$

B. IMAGE PROCESSING ALGORITHMS

This section reviews the image processing algorithms that are used to perform object detection.

1. Image Filtering

Image filtering aims to eliminate undesirable characteristics found in the raw images taken by the sensors. Jain, Kasturi, and Schunck (1995) described histogram modification as stretching the contrast of images that had intensity values within a small range, to aid in subsequent image processing efforts such as threshold selection. Linear filters are effective in removing Gaussian (or normal distribution) noise that are introduced into images by the sensor electronics, and are implemented by using the weighted sum of pixels in successive windows (Jain, Kasturi, and Schunck 1995). Figure 9 shows the implementation of a 3pix-by-3pix median filter.

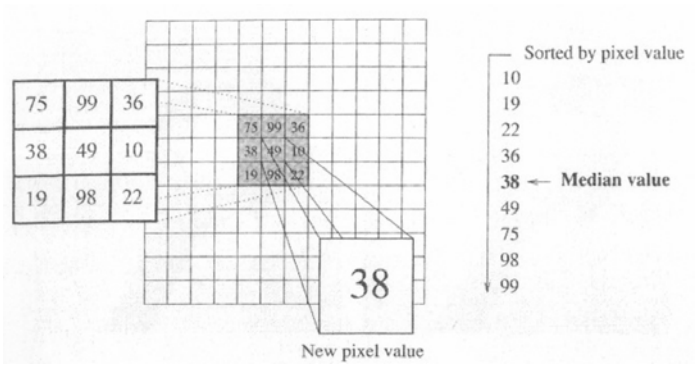


Figure 9. The median filter in a 3pix-by-3pix neighborhood.
Source: Jain, Kasturi, and Schunck (1995).

An example of the linear filter is the mean filters where each pixel is replaced by the average of all the values in the local neighborhood. The Gaussian filter is a type of linear filter with weights determined according to the shape of a Gaussian function. The median filter differs from the mean filter in that each pixel is replaced with the median of

the gray values in the local neighborhood, and that the value is not a weighted sum of the pixels.

2. Image Alignment

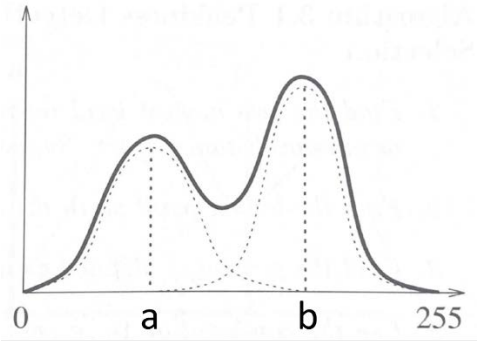
Images taken at different times for the same area of interest would invariably exhibit differences geometrically. The affine transformation is employed to align the new image (x_{new}, y_{new}) to the original image ($x_{original}$, $y_{original}$) through a combination of translation, scaling, shearing and rotation in a matrix **A** (Yakimenko 2018).

$$\begin{bmatrix} x_{original} \\ y_{original} \\ w \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} \tag{4}$$

With matching pairs of control points, the coefficients of matrix **A** could be found. Applications of image alignment are wide and varied. One example is in the alignment of medical images such as X-ray scans, so that the progress of medical diagnosis could be tracked and monitored (Wang and Chen 2013).

3. Threshold Segmentation

Threshold segmentation is the grouping of pixels that correspond to an object and the separation of other pixels from the group (Jain, Kasturi, and Schunck 1995). This is done throughout the image such that there may be multiple groups of pixels. This is shown in Figure 10, where the intensities of the background and objects have different peaks with some overlap. The threshold is selected from between the two peak intensities (points a and b) such that the object could be differentiated from the background.



The threshold value would be selected from between these two peaks. Adapted from Jain, Kasturi, and Schunck (1995).

Figure 10. Example of intensity distribution of an image with two peaks representing background (a) and object (b).

Threshold segmentation methods are used in image analysis to determine the level of vegetation (Xie et al. 2011), as well as in determining the progress of medical diagnosis as discussed in the previous section.

4. Edge Detection

Object detection in an image relies heavily on edge detection. Edges of objects are very useful in being able to detect them as they form a natural boundary with the surrounding, and they are characterized by significant local changes in the image (Jain, Kasturi, and Schunck 1995). This could be implemented by using determining the gradient **G** of the image function, where

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{5}$$

The magnitude of the gradient is given by

$$\|\mathbf{G}\| = \sqrt{G_x^{\,2} + G_y^{\,2}} \tag{6}$$

and the direction of the gradient is given by

$$\angle \mathbf{G} = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{7}$$

The Roberts operator approximates the magnitude of the gradient to

$$G_R \approx |G_x| + |G_y| \tag{8}$$

The Sobel and Prewitt operators approximates the gradient magnitude

$$G_s \approx \sqrt{{s_x}^2 + {s_y}^2} \tag{9}$$

where

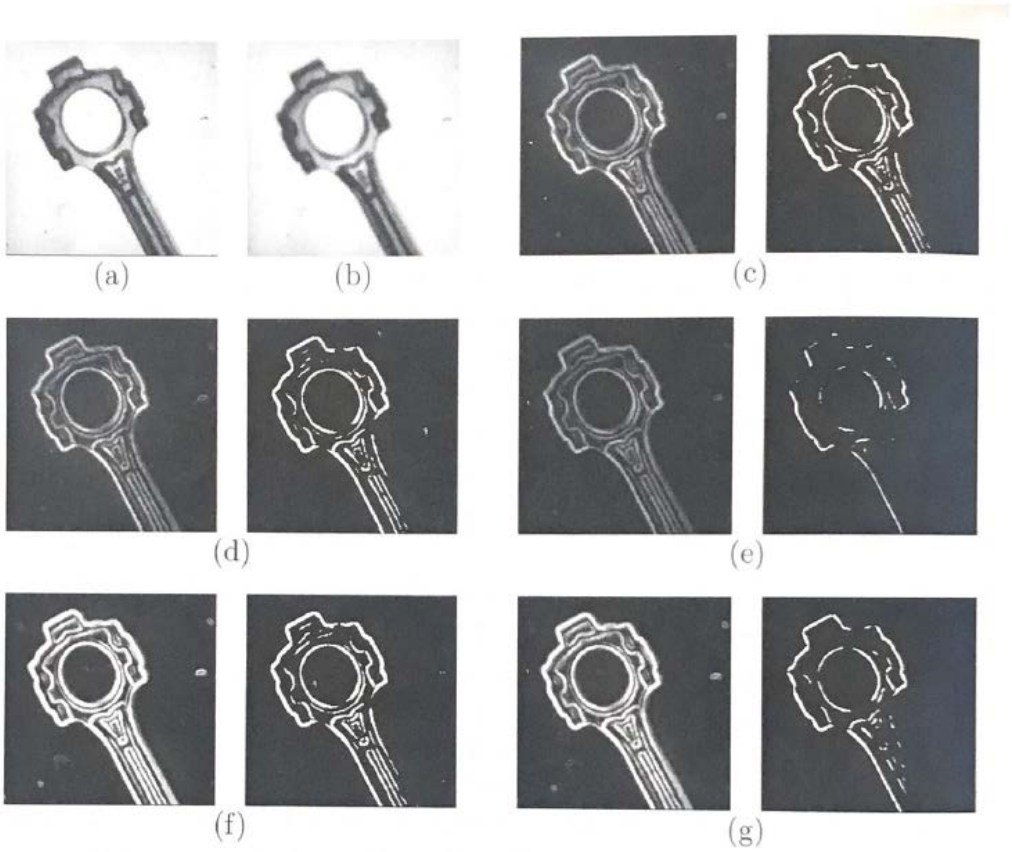
$$\begin{aligned} s_x &= (a_2 + ca_3 + a_4) - (a_0 + ac_7 + a_6) \\ s_y &= (a_0 + ca_1 + a_2) - (a_6 + ac_5 + a_4) \end{aligned} \tag{10}$$

In the partial derivatives in equation 10, the constant c is equal to 2 for the Sobel operator, while c is equal to 1 for the Prewitt operator. Figure 11 shows the labels of the neighborhood of pixels at position $[i, j]$ for the partial derivatives.

a_0	a_1	a_2
a_{07}	$[i, j]$	a_3
a_6	a_5	a_4

Figure 11. Pixel labels in Sobel and Prewitt operator.
Source: Jain, Kasturi, and Schunck (1995).

Jain, Kasturi, and Schunck (1995) illustrated the detection of a tool using different edge detectors, shown in Figure 12. 12(a) and (b) show the original image and the filtered image respectively, and 12(c) shows the detection using gradient with 1 x 2 and 2 x 1 masks on the two images, with threshold T at 32. 12(d) shows the detection using gradient with 2 x 2 masks, T at 64. 12(e) shows the detection using the Roberts operator, T at 64, 12(f) with Sobel operator, T at 225, and 12(g) with Prewitt operator, T at 225.



(a) and (b) show the original image and the filtered image respectively. (c) Gradient using 1 x 2 and 2 x 1 masks, T = 32. (d) Gradient using 2 x 2 masks, T = 64. (e) Roberts cross operator, T = 64. (f) Sobel operator, T = 225. (g) Prewitt operator, T = 225. Source: Jain, Kasturi, and Schunck (1995).

Figure 12. Detection of tool with different edge detectors.

5. Image Processing Procedure

Yuan et al. (2015) described the sequence of image processing techniques used in fire detection and tracking algorithms, where the blob is representative of a hot spot. See Figure 13.

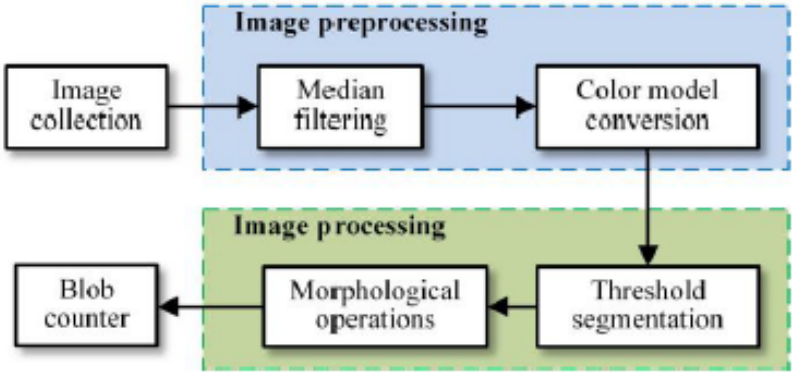


Figure 13. Sequence of image processing in fire detection and tracking. Source: Yuan et al. (2015).

This thesis adopts the same techniques with one rearrangement in the sequence. Instead of performing the threshold segmentation after the color model conversion, the morphological operations are performed on the image first.

IV. ALGORITHM DEVELOPMENT AND DATA GATHERING

This chapter describes the methodology to address the problem. To study requirements of a system of UAS that could perform airfield FOD inspection, a five-step process was adopted.

Step 1. Development of a concept of operations and GUI prototype

Step 2. Development of MATLAB code to mark out the foreign objects in the image scene

Step 3. Vary experiment parameters to determine configuration of UAS for airfield inspection

Step 4. Execution of experiments at Camp Roberts to collect the image data and process with the MATLAB code

Step 5. Data analysis and recommendations

A. CONCEPT OF OPERATIONS AND GUI PROTOTYPE

The development of the concept of operations and the GUI prototype were discussed in Chapter II. To recap, the concept of operations involves a UAS swarm lined up at one end of the runway and traversing along the length of the runway at the operator-specified altitude. The commands to control the UASs' altitude, speed, area of inspection, and the camera system used were controlled through the GUI which could be located at the control tower.

As the UASs traverse the runway, they would wirelessly transmit the images taken to the central processor for image processing and object detection. If any FOD were detected, the operator would be informed through the GUI with the image of the FOD and the location.

B. DEVELOPMENT OF MATLAB CODE

MATLAB is a programming tool that integrates computation, visualization and programming in a user-friendly computing environment (Yakimenko 2011). The

availability of modular toolboxes suitable for varied uses makes MATLAB a highly versatile and powerful tool for both academic and industry uses. In this thesis, this author made use of the Image Processing Toolbox and Computer Vision System Toolbox in MATLAB, which together allowed the use of powerful image handling, transformation, filtering and other processes by simply typing single function commands.

To allow the computer to recognize and mark out the foreign object from the image scene, requires the sequential performance of several image processing and computation steps. The computer does not understand a command such as "detect a foreign object on the runway," which an airfield inspector would. A programmer would need to break down the command into several sub-commands which the computer could perform. For this thesis, these commands are broken down into the following steps:

1. Read background image file.
2. Read current image file.
3. Convert and filter color background and images into grayscale.
4. Align current image to background image.
5. Find absolute difference of current image from background.
6. Filter the results to sharpen the resultant binary image.
7. Draw bounding boxes around the image objects.
8. Overlay bounding boxes on the current image file.

Refer to Appendix B for the MATLAB code.

C. VARY EXPERIMENT PARAMETERS

A set of four parameters were varied to determine the configuration of the UAS that could perform the FOD detection. They were 1) object color, 2) object size, 3) UAS altitude, and 4) filter window size parameter.

1. Object Color

The color of the object affects its contrast in the image being processed. The color that required the higher performance would be used to set the requirements for the system of UAS.

2. Object Size

Increasing the size of the object in the image, while keeping all other factors a constant, would result in an increase in the number of pixels for that object in the image. By increasing the number of pixels, the probability of detection for that object is higher. In this thesis, we made use of objects of different sizes, to determine the number of pixels required for the object to be detected by the algorithm. These objects were selected from the list that the FAA stipulated in AC 150/5220-24 for object detection performance. The objects described in the circular ranged from 3.1 cm wide to as much as 20 cm long. In this thesis, the objects were flat squares with sides that ranged from 3 cm to 20 cm long.

3. UAS Altitude

The altitude of the UAS would similarly affect the number of pixels for an object in the image. The higher the UAS was at, the lower the number of pixels present for each object.

4. Image Filter Window Size

The median filter was used to remove noise in the results. The default setting in MATLAB for the 2-D median filter was a 3pix-by-3pix box. When the filter parameter was increased, the number of false detections decreased correspondingly, as the filter takes the average of a larger area and applies it to each pixel. This would however also reduce the likelihood of detecting a small object as it would be overwritten by the average value of the pixels surrounding the object.

D. EXECUTION OF EXPERIMENTS AT CAMP ROBERTS

Camp Roberts is in San Miguel, California, and is located approximately 100 miles southeast of the NPS campus in Monterey, California. It is a National Guard base, with

facilities including billeting and logistics support to military personnel and units on training (California Military Department 2018). The Naval Postgraduate School performs its field experimentation of unmanned systems at the McMillan Airport within Camp Roberts, and that was where the experiments in this thesis was conducted over two occasions on April 27, 2018, and 12-13 July, 2018. Figure 14 shows the author's experimental set-up at Camp Roberts.

Figure 14. Experimental set-up at Camp Roberts

1. System Specifications

The specifications of the UAS, camera and computer used in the experiment are described in this section.

a. UAS Specifications

The UAS used in the experiment was a DJI Inspire 1, with relevant specifications listed in Table 3.

Table 3. UAS and remote controller specifications

UAS	
Weight	3060 g (6.74 lbs)
Max Speed	79 kph (49 mph)
Max Wind Speed Resistance	10 m/s
Max Flight Time	18 min
Diagonal Distance (Dimensions)	581 mm (22.8”) in Landing Mode
Remote Controller	
Max Transmitting Distance	Up to 5 km (3.1 miles) when FCC compliant
Video Output Ports	USB, mini HDMI

The remote controller was connected to this author’s iPhone 7, with the DJI Go and Pix4Dcapture apps installed.

b. Camera Specifications

The DJI’s proprietary Zenmuse X3 was used for image capturing. The specifications of the camera are listed in Table 4.

Table 4. Zenmuse X3 camera specifications

Sensor Type	Complementary Metal-Oxide-Semiconductor (CMOS)
Sensor Dimensions	6.17 x 4.55 mm
Shutter Speed	1/8000 sec
Focal Length (35 mm Equivalent)	20 mm
Aperture	F/2.8
FOV	94°
Max Pixels	12.4 million
Resolution	16:9 - 4000 x 2250 4:3 - 4000 x 3000

c. Computer Specifications

The computer that was used to process the images was a MSI GS43VR with 14” built-in monitor. The specifications are listed in Table 5.

Table 5. Image processing computer specifications

Processor	Intel i7-7700 2.80 GHz
RAM	16 GB
Hard Disk Drive	128 GB m.2 SSD 1 TB SATA HDD
Screen Resolution	1920 x 1080
Operating System	Windows 10 Home (64-bit)
Image Processing Software	MATLAB R2018a Update 2

While the computer was equipped with a graphics co-processor, the MATLAB algorithm did not employ it in the image processing.

2. Test Scenario

The scenario for the test simulates two different time slices of the runway inspection process performed by the system of UAS. During the setup or calibration, the system of multiple UAS would be traversing the length of the runway at a certain altitude above the runway. While traversing, each UAS would stop at regular intervals to take images of the runway, which would be stored as the background image. The calibration process is complete when the images of the entire runway have been taken and stored in the image server. During operations, the system of UAS would again be traversing the runway at the same altitude as before, and taking images at regular intervals. During this time, we expect to detect foreign objects on the runway. The images taken would be transferred to the image server where the image detection algorithm would be used to detect the foreign objects in the image. The instances in time when the background image and the corresponding operational image were taken are the time slices simulated in the experiment.

3. Measures of Performance

The measures of performance in this experiment are the variables. To recap, the variables are as follows:

1. Object color

2. Object size

3. UAS altitude

4. Image filter window size

The objective is for MATLAB to automatically draw the bounding boxes around the objects that were introduced into the image. When the bounding boxes are drawn, it is assumed that the objects are detected by the algorithm.

The system configuration that would be eventually selected would be the system configuration highest detection rate at the highest altitude and the largest filter size, for the smallest object. To ensure that the system could perform with different contrast levels, and results from the color that returned the lower detections would be used. See Table 6.

Table 6. Desired metrics

UAS altitude	Highest
Image filter window size	Largest
Object size	Smallest
Object color	More stringent

4. Data Collection Procedures

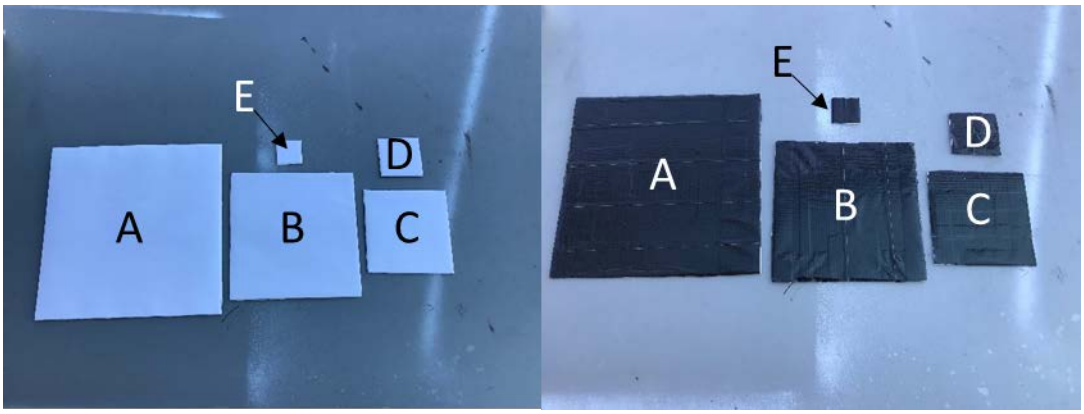
The images used in this study were taken at the runway access adjacent to the displaced threshold for Runway 28. See Figure 15.

Figure 15. Images in this study taken at the access circled in red. Adapted from Google Maps (2018).

a. Foreign Objects

The "foreign objects" in the study were made using square pieces of cardboard, with alternating sides of white and black. There were a total of five objects, with the following dimensions. See Figure 16.

1. Object A - 20 cm x 20 cm
2. Object B - 15 cm x 15 cm
3. Object C - 10 cm x 10 cm
4. Object D - 5 cm x 5 cm
5. Object E - 3 cm x 3 cm



Identifying alphabets are not present in actual specimens.

Figure 16. Foreign objects with alternating white (left) and black (right) sides in image to be detected.

b. Experimental Set-up

Red duct tape was used to bracket the area where the objects were placed. The duct tape also served as markers which the algorithm could use to align the images, so that alignment could be eliminated as a form of variation in this study. To further support the alignment of the image pair, the area where the images were taken included the taxiway centerline. See Figure 17.

Figure 17. Red duct tape and taxiway centerline bracketed the area where the objects were placed

c. Image Collection

The background image was taken using the DJI Inspire 1 UAS. To support the image capturing, the Pix4Dcapture app in "Free Flight" mode was used to automatically take pictures at 1 m intervals from 1 m to 10 m above ground level (AGL), while the UAS was manually flown to 10 m from the ground, at a vertical speed of between 0.2 m/s and 0.3 m/s. Then each object was placed between the red duct tape, and images were taken again at altitudes of 1 m interval from 1 to 10 m. This process was repeated for the five objects with the white side facing up, and then again with the black side facing up. Figure 18 shows the author performing an image collection test at Camp Roberts.

Figure 18. Image collection at Camp Roberts

d. Image Processing

After the images were taken, there were downloaded onto this author's laptop. The images were then batch processed on this author's laptop using MATLAB, with the "white" objects first followed by the "black" objects, while increasing the filter sizes in steps of 2 pixels from 3 x 3 to 25 x 25 for each color-size-altitude combination. The total number of data points was 1200 from all the variations. The processing took a total of 11 hours, with each image taking approximately 6 minutes to be processed. Figure 19 shows an example of the image processing output, with the left image showing the foreign object detected with the bounding box drawn around it. The presence of other boxes indicated

that false detections were made. Some of the false detections could be attributed to image alignment or the UAS shadow.

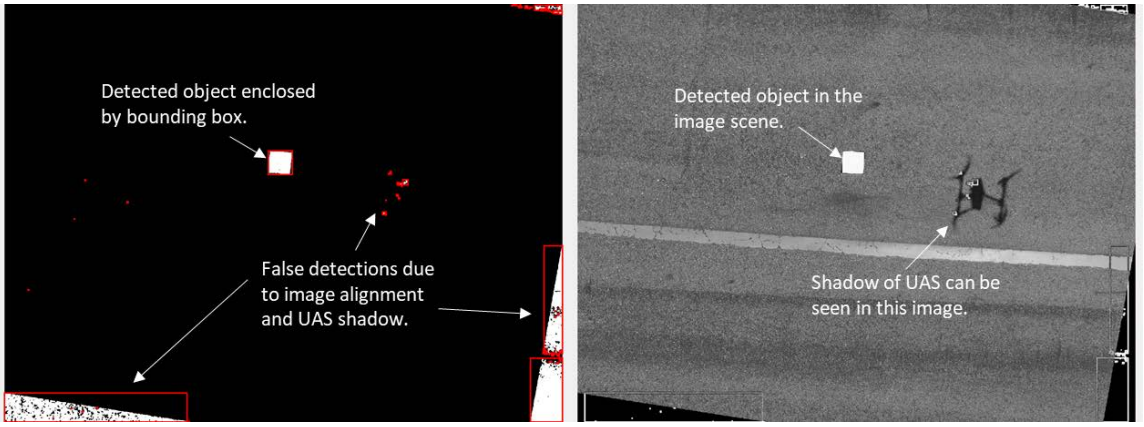


Figure 19. Examples of the image processing output

e. Tabulate Results

The results of the object detection were tabulated in Microsoft Excel. An example is shown in Table 7.

Table 7. Example of object detection results. (Y) represented object detected, and (N) represented object not detected in the last column.

Object Dimensions (cm)	Object Color	Filter Parameter	UAS Altitude (m)	Object Detected
3	Black	3	1	Y
3	Black	5	1	Y
3	Black	7	1	Y
3	Black	9	1	Y
3	Black	11	1	Y
3	Black	13	1	Y
3	Black	15	1	Y
3	Black	17	1	N
3	Black	19	1	N
3	Black	21	1	N
3	Black	23	1	N
3	Black	25	1	N
3	Black	3	2	Y

Object Dimensions (cm)	Object Color	Filter Parameter	UAS Altitude (m)	Object Detected
3	Black	5	2	Y
3	Black	7	2	Y
3	Black	9	2	Y
3	Black	11	2	Y
3	Black	13	2	N
3	Black	15	2	N
3	Black	17	2	N
3	Black	19	2	N
3	Black	21	2	N
3	Black	23	2	N
3	Black	25	2	N

5. Data Collection Challenges

The collection of the data proved to be challenging, despite the availability of a sanitized airspace for conducting the research. Wind conditions at McMillan Airport could be strong particularly in the afternoons. It was a challenge to maintain the UAS over the specimen FOD particularly at low altitudes, as the UAS tended to drift away relative to the FOV. This problem also extended to horizontal flights covering an extended area, as the UAS drifted away from the direction of travel in the presence of crosswinds.

Keeping the climb and descent rate of the UAS stable was also a challenge in fluctuating winds, which resulted in difficulties to take images at the specific altitude intervals required in this thesis. While the images were eventually taken during a short break in the winds, control of the UAS particularly in high or fluctuating winds need to be improved in future work to ensure the feasibility of the AFDS.

V. DATA ANALYSIS

A. EFFECTS OF FOD PARAMETERS

The results of the field tests on changing FOD parameters are discussed in this section.

1. Effects of Object Color

The objects that were white were able to be detected more often that the objects in black. Of the 600 detection opportunities in each color, the black objects were detected 215 times, or 35.8% of the time, while the white objects were detected 548 times or 91.3% of the time.

2. Effects of Object Size

As the object size increased, the detection increased. For the black objects, the smallest object (3 cm x 3 cm) was detected 34 times out of 120 times, or 28.3% of the time, while the largest object (20 cm x 20 cm) were detected 49 times, or 40.8% of the time. For the white objects, the smallest object was detected 83 times, or 69.2% of the time, while the objects that were 10 cm or larger were detected 100% of the time. See Figure 20.

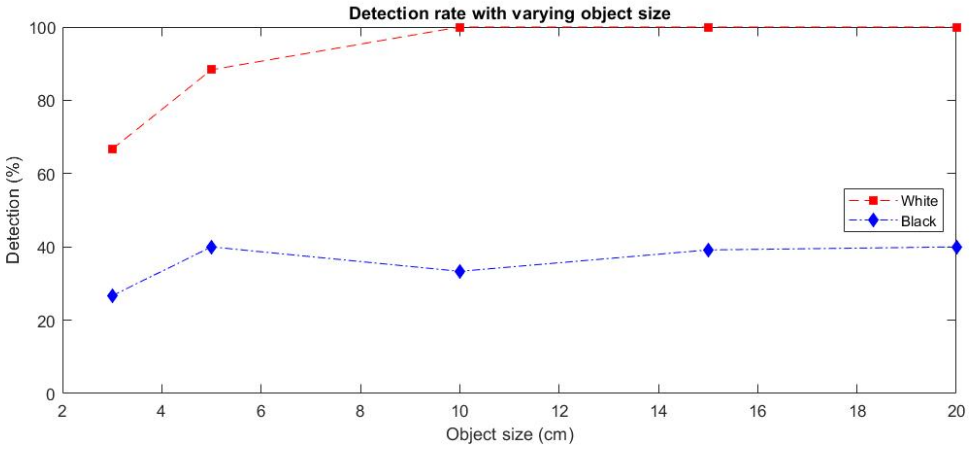


Figure 20. Detection rate of varying object size for black and white objects

3. Effects of UAS Altitude

As the altitude of the UAS increased, the detection of the object decreased. For black objects, the detection rate was 58 times out of 60, or 96.7% at 1 m, which decreased to 6 times out of 60, or 10% at 10 m. For white objects, the detection rate was 100% at 1 m, which decreased to 47 times or 78.3% at 10 m. See Figure 21.

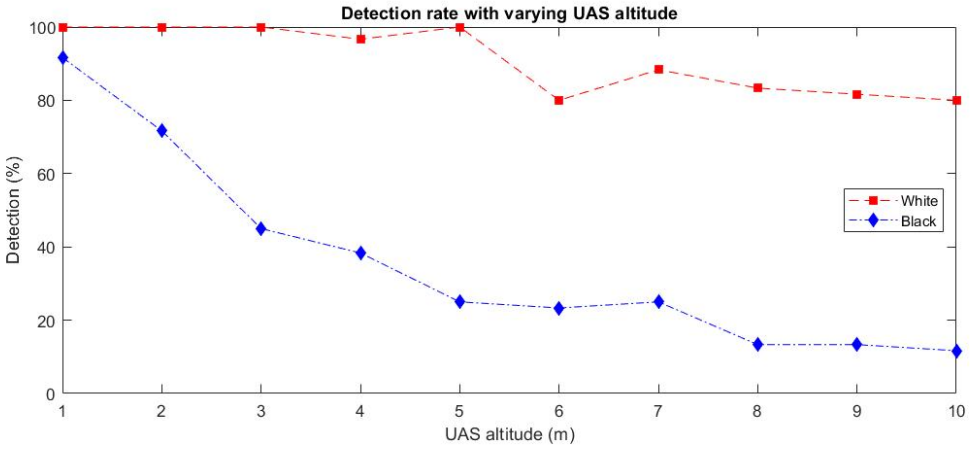


Figure 21. Detection rate of varying UAS altitude for black and white objects

4. Effects of Filter Size

As the filter size increased, the detection of the object decreased. For black objects, the detection rate was 49 times out of 50, or 98% at the default 3 x 3 median filter, which decreased to 5 times, or 10% with the 25 x 25 median filter. For white objects, the detection rate was 100% at the default 3 x 3 median filter, and that decreased to 39 times, or 78% with the 25 x 25 median filter. See Figure 22.

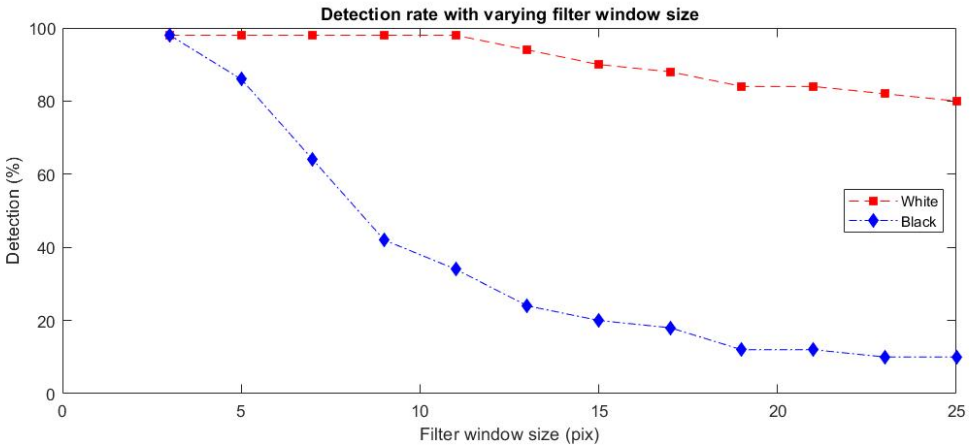


Figure 22. Detection rate of varying filter size for black and white objects

B. SELECTION OF AFDS PARAMETERS

The configuration for the AFDS would be based on the detection of the black objects, as it was found to be more challenging to detect black as opposed to white objects.

For the combination of altitude and filter size, there was a set of configurations that allowed for the objects to be detected. See Table 8 for the detection rate of the black objects for each UAS altitude-filter window size configuration.

Table 8. Detection rate of black objects in the UAS altitude-filter window size coordinates

		UAS altitude AGL (m)									
		1	2	3	4	5	6	7	8	9	10
Filter window size (pix)	3	100%	100%	100%	100%	100%	100%	100%	100%	100%	80%
	5	100%	100%	100%	100%	100%	100%	100%	80%	40%	40%
	7	100%	100%	100%	100%	60%	80%	20%	40%	0%	0%
	9	100%	100%	100%	60%	20%	0%	20%	0%	0%	0%
	11	100%	100%	80%	40%	0%	0%	0%	0%	0%	0%
	13	100%	80%	60%	40%	0%	0%	0%	0%	0%	0%
	15	100%	80%	0%	0%	0%	0%	0%	0%	0%	0%
	17	100%	80%	0%	0%	0%	0%	0%	0%	0%	0%
	19	100%	60%	0%	0%	0%	0%	0%	0%	0%	0%
	21	100%	40%	0%	0%	0%	0%	0%	0%	0%	0%
	23	80%	20%	0%	0%	0%	0%	0%	0%	0%	0%
	25	80%	20%	0%	0%	0%	0%	0%	0%	0%	0%

To reduce the likelihood of false detections, it was proposed to a filter window size of no smaller than 7 pixels. Hence, the UAS would need to be flown at 4 m from the ground to be able to detect objects that have dimensions of 3 cm or smaller. Table 9 shows the summary of the proposed AFDS configuration.

Table 9. Proposed AFDS configuration

UAS Altitude	4 m
Image filter window size	7pix-by-7pix

1. Ground Sample Distance

For an object of 3 cm dimension, the altitude at which the object could be detected with the Zenmuse X3 was 4 m. Using equation (1), the real focal length of the camera would be

$$F_R[mm] = \frac{20 \times 6.16}{34.6}$$
$$F_{R,X3}[mm] = 3.561 \text{ mm} \tag{11}$$

From equation (2), the ground distance covered by the UAS at 4 m is given by

$$D_W = \frac{4000 \times 6.16}{3.561}$$
$$D_W = 6920 \text{ mm} \tag{12}$$

The ground width of each image is 6920 mm or 6.92 m. From equation (3), The GSD of the image is

$$GSD = \frac{D_W}{\text{Image Width}}$$
$$GSD = \frac{6920}{4000} = 1.73 \text{ mm/pixel} \tag{13}$$

2. Number of UASs Required

With the ground width of each image being 6.92 m, the number of UAS required would be the width of the runway divided by the ground width of each image. With a 10% overlap on either side of each UAS, the coverage of each UAS becomes 5.54 m. Hence, a runway that is 60 m wide (such as Runway 02L/20R at the Singapore Changi Airport) would require 11 UAS spaced 5.54 m apart. A narrower runway that is 150 ft or 45.72 m wide (such as Runway 10R/28L at Monterey Regional Airport) would require 9 UAS similarly spaced apart. See Figure 23.

3. Number of Images Required

For a 4000 x 3000 pixel image covering a width of 6.92 m on the ground, the image would cover a distance of 5.19 m. With a 10% overlap for the images, the distance covered by each image would be 4.15 m. For a runway that is 4000 m long at the Singapore Changi Airport, each UAS would take 964 images to cover the length. For the runway at Monterey Regional Airport that is 7175 ft or 2187 m long, each UAS would take 527 images to cover the length of the runway. See Figure 23.

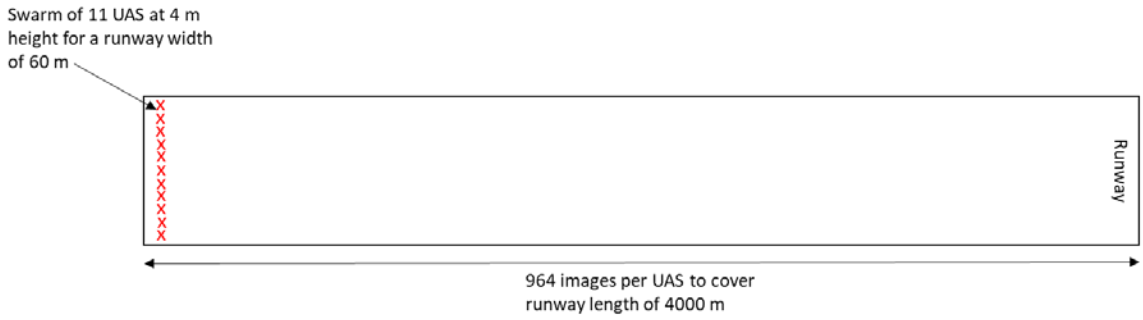


Figure 23. For inspection at UAS altitude of 4 m, 11 UAS are required and 964 images per UAS would be taken to cover a runway length of 4000 m.

4. Data Generation and Transfer Rate

The data size of one 4000 x 3000 pixel image is approximately 5 MB in JPEG form. The amount of data generated per run for a 4000 m x 60 m runway would be 10604 images x 5 MB for a total data size of 53 GB. The amount of data generated for a smaller runway

that is 2187 m x 46 m would be 24 GB. Assuming that the UAS travels at 5 m/s including the time to stop and take the images, it would take 800 seconds or 13.3 minutes to cover a 4000 m runway, and 437 seconds or 7.3 minutes to cover a 2187 m runway.

To maximize the time for image processing, the data should be transferred as soon as the images are taken. Hence, the data transfer rate should be

$$Bitrate = \frac{53GB \times 8bit}{800}$$
$$Bitrate = 0.53\text{Gbps} \tag{14}$$

5. Processing Time

Each image took 360 seconds or 6 minutes to be processed in MATLAB on this author's laptop computer. For a lane of 964 images, that would take 5784 minutes, or 96.4 hours. For compiled code, this would be reduced to approximately 3.6 seconds per image, or slightly less than an hour. Such a length of time is too slow to be practical in the operational environment. For the system to be practical, the processing time should ideally be limited to less than 30 minutes. That means that processing speeds need to be increased by a factor of 2 or better, which could be accomplished by code optimization, or performing the processing on a more powerful computer, or both.

6. Summary of System Specifications and Cost

For a system of UAS capable of performing inspection of a 4000 m x 60 m runway, the costs and quantity are listed in Table 10. This assumes a parallel computing system with each desktop computer processing the inputs from each UAS. Only the hardware costs are included in this analysis.

Table 10. System specifications and cost of AFDS

Item	System	Quantity	Unit Cost	Cost
Processor	Desktop Computers	11	$3,000	$33,000
UAS	DJI Inspire 1 with Zenmuse X3	11	$2,000	$22,000
Modem	AC5300-standard Router	1	$500	$500
			Total	**$55,500**

VI. CONCLUSION AND RECOMMENDATIONS

A. SUMMARY

This thesis presented the requirements of a system of Group 1 UAS swarm to perform FOD management at an airport. The concept of operations of the UAS swarm was first presented, together with the GUI to support the operations. The prototype GUI was then developed and tested and found to be feasible.

Field-testing was conducted at Camp Roberts with a consumer-class DJI Inspire 1 UAS and a typical 4K 94° FOV Zenmuse X3 camera to collect the required images to perform the image analysis. These tests also provided the opportunity to conduct preliminary assessment of the challenges that need to be overcome to implement the proposed technology.

The feasibility assessment of the computation algorithms for image processing was performed on a consumer-class laptop with MATLAB. Object parameters including the size and color, as well as the altitude of the UAS and the image processing window filter size were varied to determine the recommended configuration for the system.

Finally, the performance of the system was evaluated based on the recommended configuration of the system. With the system specifications used in the experiment, the processing time was assessed to be too long to be practical. However, with code optimization and better hardware, FOD management with consumer-class Group 1 UAS and computing equipment could be implemented in the near future. This thesis found that the concept is feasible, with more testing required to ensure the reliability of the system.

B. RECOMMENDATIONS FOR FUTURE WORK

Through the development and testing of AFDS, the future work is recommended as follows:

- Develop image extraction from video images to improve image collection speeds. The method used in this thesis entailed the UAS stopping at regular intervals over the runway to take still images.

- Improve UAS control in high and fluctuating wind conditions.
- Develop UAS control using the GUI.
- Optimize image-processing code to speed up object detection.
- Develop and test the data transfer infrastructure from the UAS swarm to the image processor. Together with the UAS control GUI, these two developments would enable the implementation of the full end-to-end AFDS system.
- Develop system redundancies should any UAS fail during the inspection process.

Future development in this concept would provide further proof of the feasibility of this system, and hopefully result in the eventual implementation of the AFDS in an operational environment. Acceptance of unmanned technologies such as the AFDS would improve safety in airport operations, while providing the opportunity for productivity growth in manpower-limited countries like Singapore.

APPENDIX A. MATLAB CODE FOR GRAPHICAL USER INTERFACE

```
classdef AFDS2 < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                          matlab.ui.Figure
        CoverageButtonGroup               matlab.ui.container.ButtonGroup
        FullSweepButton                   matlab.ui.control.RadioButton
        PartialSweepButton                matlab.ui.control.RadioButton
        InspectionDistancemEditFieldLabel  matlab.ui.control.Label
        InspectionDistancemEditField      matlab.ui.control.NumericEditField
        AirfieldMapPanel                  matlab.ui.container.Panel
        WindEditFieldLabel                matlab.ui.control.Label
        WindEditField                     matlab.ui.control.NumericEditField
        WindDirection                     matlab.ui.control.EditField
        UIAxes                            matlab.ui.control.UIAxes
        UASDisplayPanel                   matlab.ui.container.Panel
        UIAxes2                           matlab.ui.control.UIAxes
        DistanceScannedmEditFieldLabel    matlab.ui.control.Label
        DistanceScannedmEditField         matlab.ui.control.NumericEditField
        InspectSwitchLabel                matlab.ui.control.Label
        InspectSwitch                     matlab.ui.control.ToggleSwitch
        CameraPanel                       matlab.ui.container.Panel
        ModelDropDownLabel                matlab.ui.control.Label
        ModelDropDown                     matlab.ui.control.DropDown
        FOVdegreesEditFieldLabel          matlab.ui.control.Label
        FOVdegreesEditField               matlab.ui.control.NumericEditField
        MinObjectSizemmEditFieldLabel     matlab.ui.control.Label
        MinObjectSizemmEditField          matlab.ui.control.NumericEditField
        UASAltitudemSpinnerLabel          matlab.ui.control.Label
        UASAltitudemSpinner               matlab.ui.control.Spinner
        UASEditFieldLabel                 matlab.ui.control.Label
        UASEditField                      matlab.ui.control.NumericEditField
        ImagesperLaneEditFieldLabel       matlab.ui.control.Label
        ImagesperLaneEditField            matlab.ui.control.NumericEditField
        ResolutionEditFieldLabel          matlab.ui.control.Label
        ResolutionEditField               matlab.ui.control.EditField
        InspectioninProgressLampLabel     matlab.ui.control.Label
        InspectioninProgressLamp          matlab.ui.control.Lamp
        AirportRunwayPanel                matlab.ui.container.Panel
        AirportDropDownLabel              matlab.ui.control.Label
        AirportDropDown                   matlab.ui.control.DropDown
        RunwayDropDownLabel               matlab.ui.control.Label
        RunwayDropDown                    matlab.ui.control.DropDown
        LengthmEditFieldLabel             matlab.ui.control.Label
        LengthmEditField                  matlab.ui.control.NumericEditField
```

```
    WidthmEditFieldLabel            matlab.ui.control.Label
    WidthmEditField                 matlab.ui.control.NumericEditField
    DateEditFieldLabel              matlab.ui.control.Label
    DateEditField                   matlab.ui.control.EditField
    MaxSpeedEditFieldLabel          matlab.ui.control.Label
    MaxSpeedEditField               matlab.ui.control.NumericEditField
    UASSpeedEditFieldLabel          matlab.ui.control.Label
    UASSpeedEditField               matlab.ui.control.NumericEditField
    SpeedmsSliderLabel              matlab.ui.control.Label
    SpeedmsSlider                   matlab.ui.control.Slider
    ResetDistanceScannedButton      matlab.ui.control.Button
end

properties (Access = private)
    Windspeed % Description
    Rwy_len
    Rwy_wd
    UAS_alt
    UAS
end

methods (Access = private)


    function UAS = UAS_Num(app,l,n,m,cam) % Calculate # UAS required
        % n represents runway width, m represents UAS altitude
        % cam represents camera model, l represents runway length

        if strcmp(cam,'Zenmuse X3')
            eq_len = 20;
        elseif strcmp(cam,'Zenmuse Z3')
            eq_len = 22;
        end

        Fr = (6.16*eq_len)/34.6;
        Dw = (m*1000*6.16) / Fr;
        D_ht = Dw * 3/4;
        app.ImagesperLaneEditField.Value = ceil(l/(D_ht*0.8/1000));
        Overlap = Dw * 0.1;
        Scan_width = Dw - (2*Overlap);
        UAS = ceil((n*1000)/Scan_width);
        GSD = Dw / 4000; % 4000 = # pixels in width
        app.MinObjectSizemmEditField.Value = (30/1.73) * GSD;

    end


    function sc = screen(app) % UAS display
```

```
            sc = true(399,231);
            for m = 1:100
                x = round(399*rand+1);
                y = round(231*rand+1);
                sc(x,y) = false;
            end
        end

        function [windspeed,direction] = wind(app)
            windspeed = rand*40;
            di = {'N','S','E','W'};
            r = ceil(rand*4);
            direction = di{r};
        end

        function fod = fod_found(app,p)
            likely = rand;
            if likely < p
                fod = 1; %fod found
            else
                fod = 0; %fod not found
            end
        end
    end
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app)
            [spd,di] = wind(app);
            app.WindEditField.Value = spd;
            app.WindDirection.Value = di;


            t = datetime('now');
            formatOut = 1;
            app.DateEditField.Value = datestr(t,formatOut);
            app.MaxSpeedEditField.Value = 22;
            app.UASSpeedEditField.Value = app.SpeedmsSlider.Value;

            app.RunwayDropDown.Items = {'02L/20R','02C/20C'};
            app.Rwy_len = 4000;
            app.Rwy_wd = 60;
            app.WidthmEditField.Value = app.Rwy_wd;
            app.LengthmEditField.Value = app.Rwy_len;
            app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value;
            app.UAS_alt = app.UASAltitudemSpinner.Value;
            selectedButton = app.CoverageButtonGroup.SelectedObject;
```

```
            app.ModelDropDown.Value = 'Zenmuse X3';
            app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                app.UAS_alt,app.ModelDropDown.Value);
            app.FOVdegreesEditField.Value = 94;


            %https://www.mathworks.com/matlabcentral/answers/360670-
imshow-in-app-designer-image-size-doesn-t-fit
            % Airfield Map
            % Fill figure with axes and remove tick labels

            app.UIAxes.Position = [32 56 451 621];

            % Remove title, axis labels, and tick labels
            title(app.UIAxes, []);
            xlabel(app.UIAxes, []);
            ylabel(app.UIAxes, []);
            app.UIAxes.XAxis.TickLabels = {};
            app.UIAxes.YAxis.TickLabels = {};
            % Display image and stretch to fill axes
            I = imshow('WSSS.png', 'Parent', app.UIAxes, ...
                'XData', [1 app.UIAxes.Position(3)], ...
                'YData', [1 app.UIAxes.Position(4)]);
            % Set limits of axes
            app.UIAxes.XLim = [0 I.XData(2)];
            app.UIAxes.YLim = [0 I.YData(2)];

            % Fill figure with axes and remove tick labels
            %app.UIAxes2.Position = [12 10 368 200];
            app.UIAxes2.Position = [12 9 399 231];

            % Remove title, axis labels, and tick labels
            title(app.UIAxes2, []);
            %xlabel(app.UIAxes2, []);
            %ylabel(app.UIAxes2, []);
            app.UIAxes2.XAxis.TickLabels = {};
            app.UIAxes2.YAxis.TickLabels = {};


        end
        % Value changed function: AirportDropDown
        function AirportDropDownValueChanged(app, event)
            Airport_value = app.AirportDropDown.Value;
            if strcmp(Airport_value,'Singapore Changi Airport')
                I = imshow('WSSS.png', 'Parent', app.UIAxes, ...
                    'XData', [1 app.UIAxes.Position(3)], ...
                    'YData', [1 app.UIAxes.Position(4)]);
```

```
                app.RunwayDropDown.Items = {'02L/20R','02C/20C'};
                app.Rwy_len = 4000;
                app.Rwy_wd = 60;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.FullSweepButton.Value = true;
                app.RunwayDropDown.Enable = 'on';
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;
                [spd,di] = wind(app);
                app.WindEditField.Value = spd;
                app.WindDirection.Value = di;

            elseif strcmp(Airport_value,'Monterey Regional Airport')
                I = imshow('KMRY.png', 'Parent', app.UIAxes, ...
                    'XData', [1 app.UIAxes.Position(3)], ...
                    'YData', [1 app.UIAxes.Position(4)]);
                app.RunwayDropDown.Items = {'10L/28R','10R/28L'};
                app.Rwy_len = 1068;
                app.Rwy_wd = 18;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.FullSweepButton.Value = true;
                app.RunwayDropDown.Enable = 'on';
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;
                [spd,di] = wind(app);
                app.WindEditField.Value = spd;
                app.WindDirection.Value = di;

            else
                I = imshow('KSNS.png', 'Parent', app.UIAxes, ...
                    'XData', [1 app.UIAxes.Position(3)], ...
                    'YData', [1 app.UIAxes.Position(4)]);
                app.RunwayDropDown.Items = {'08/26','13/31'};
                app.Rwy_len = 1830;
                app.Rwy_wd = 46;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
```

```
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.FullSweepButton.Value = true;
                app.RunwayDropDown.Enable = 'on';
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;
                [spd,di] = wind(app);
                app.WindEditField.Value = spd;
                app.WindDirection.Value = di;

            end
        end
        % Value changed function: SpeedmsSlider
        function SpeedmsSliderValueChanged(app, event)
            Speed_value = app.SpeedmsSlider.Value;
            app.UASSpeedEditField.Value = Speed_value;
        end
        % Value changed function: RunwayDropDown
        function RunwayDropDownValueChanged(app, event)
            Runway_Length = app.RunwayDropDown.Value;
            if strcmp(Runway_Length,'02L/20R')
                app.Rwy_len = 4000;
                app.Rwy_wd = 60;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;

            elseif strcmp(Runway_Length,'02C/20C')
                app.Rwy_len = 4000;
                app.Rwy_wd = 60;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;

            elseif strcmp(Runway_Length,'10L/28R')
                app.Rwy_len = 1068;
```

```
                app.Rwy_wd = 18;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;

            elseif strcmp(Runway_Length,'10R/28L')
                app.Rwy_len = 2187;
                app.Rwy_wd = 46;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;

            elseif strcmp(Runway_Length,'08/26')
                app.LengthmEditField.Value = 1830;
                app.WidthmEditField.Value = 46;
                app.Rwy_len = 1830;
                app.Rwy_wd = 46;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;

            else
                app.Rwy_len = 1470;
                app.Rwy_wd = 46;
                app.WidthmEditField.Value = app.Rwy_wd;
                app.LengthmEditField.Value = app.Rwy_len;
                app.InspectionDistancemEditField.Value =
app.LengthmEditField.Value
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);
                app.FullSweepButton.Value = true;
```

```
            end
        end
        % Value changing function: SpeedmsSlider
        function SpeedmsSliderValueChanging(app, event)
            SpeedchangingValue = event.Value;
            app.UASSpeedEditField.Value = SpeedchangingValue;
        end
        % Callback function
        function InspectButtonValueChanged(app, event)
        end
        % Value changed function: UASAltitudemSpinner
        function UASAltitudemSpinnerValueChanged(app, event)
            app.UAS_alt = app.UASAltitudemSpinner.Value;
            app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                app.UAS_alt,app.ModelDropDown.Value);
        end
        % Value changed function: InspectSwitch
        function InspectSwitchValueChanged(app, event)
            switch_val = app.InspectSwitch.Value;
            if strcmp(app.InspectSwitch.Value,'Start')
                app.InspectioninProgressLamp.Color = [0 1 0];
                app.DateEditField.Enable = 'off';
                app.AirportDropDown.Enable = 'off';
                app.RunwayDropDown.Enable = 'off';
                app.ModelDropDown.Enable = 'off';
                app.UASAltitudemSpinner.Enable = 'off';
                app.SpeedmsSlider.Enable = 'off';
                app.FullSweepButton.Enable = 'off';
                app.PartialSweepButton.Enable = 'off';
                app.ResetDistanceScannedButton.Enable = 'off';
                app.LengthmEditField.Enable = 'off';
                app.WidthmEditField.Enable = 'off';
                app.MaxSpeedEditField.Enable = 'off';
                app.UASSpeedEditField.Enable = 'off';
                app.UASEditField.Enable = 'off';
                app.ImagesperLaneEditField.Enable = 'off';
                app.ResolutionEditField.Enable = 'off';
                app.FOVdegreesEditField.Enable = 'off';
                app.MinObjectSizemmEditField.Enable = 'off';
                app.InspectionDistancemEditField.Enable = 'off';

                k = app.DistanceScannedmEditField.Value;

                for i = k:app.InspectionDistancemEditField.Value
                    if strcmp(app.InspectSwitch.Value,'Stop')
                        %app.StartStopEditField.Value =
app.InspectSwitch.Value;
```

```
                    break
                end
                if i == app.InspectionDistancemEditField.Value
                    app.InspectSwitch.Value = 'Stop';
                    %app.StartStopEditField.Value =
app.InspectSwitch.Value;
                    %break
                end

                %app.StartStopEditField.Value =
app.InspectSwitch.Value;
                app.DistanceScannedmEditField.Value = i;

                prob = 0.001; %Likelihood of FOD
                fod = fod_found(app,prob);
                str = 'FOD Found';
                if fod == 1
                    out{i} = sprintf('%s at %d m',str,i);
                    fig_no = figure('Name',out{i});
                    fig_no.MenuBar = 'none';
                    imshow('01.JPG');
                end

                %Display image and stretch to fill axes
                J = imshow(app.screen, 'Parent', app.UIAxes2, ...
                    'XData', [1 app.UIAxes2.Position(3)], ...
                    'YData', [1 app.UIAxes2.Position(4)]);
                % Set limits of axes
                app.UIAxes2.XLim = [0 J.XData(2)];
                app.UIAxes2.YLim = [0 J.YData(2)];
                pause(1/app.UASSpeedEditField.Value)

            end
        end
        app.InspectSwitch.Value = 'Stop';
        %app.StartStopEditField.Value = app.InspectSwitch.Value;
        app.InspectioninProgressLamp.Color = [1 0 0];
        app.AirportDropDown.Enable = 'on';
        app.DateEditField.Enable = 'on';
        app.RunwayDropDown.Enable = 'on';
        app.ModelDropDown.Enable = 'on';
        app.UASAltitudemSpinner.Enable = 'on';
        app.SpeedmsSlider.Enable = 'on';
        app.FullSweepButton.Enable = 'on';
        app.PartialSweepButton.Enable = 'on';
        app.ResetDistanceScannedButton.Enable = 'on';
        app.LengthmEditField.Enable = 'on';
        app.WidthmEditField.Enable = 'on';
```

```
            app.MaxSpeedEditField.Enable = 'on';
            app.UASSpeedEditField.Enable = 'on';
            app.UASEditField.Enable = 'on';
            app.ImagesperLaneEditField.Enable = 'on';
            app.ResolutionEditField.Enable = 'on';
            app.FOVdegreesEditField.Enable = 'on';
            app.MinObjectSizemmEditField.Enable = 'on';
            app.InspectionDistancemEditField.Enable = 'on';

        end
        % Value changed function: ModelDropDown
        function ModelDropDownValueChanged(app, event)
            camera_model = app.ModelDropDown.Value;
            if strcmp(camera_model,'Zenmuse X3')
                app.FOVdegreesEditField.Value = 94;
            elseif strcmp(camera_model,'Zenmuse Z3')
                app.FOVdegreesEditField.Value = 92;
            end
            app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                app.UAS_alt,app.ModelDropDown.Value);
        end
        % Selection changed function: CoverageButtonGroup
        function CoverageButtonGroupSelectionChanged(app, event)
            Airport_value = app.AirportDropDown.Value;
            selectedButton = app.CoverageButtonGroup.SelectedObject;
            %app.CoverageEditField.Value = selectedButton.Text;
            if app.PartialSweepButton.Value == true
                close all;
                fig = figure('Name','Select Start and End
points.','NumberTitle','off');
                fig.MenuBar = 'none';
                if strcmp(Airport_value,'Singapore Changi Airport')
                    airfield = imshow('WSSS.png');
                    scale = 5.2951;
                elseif strcmp(Airport_value,'Monterey Regional Airport')
                    airfield = imshow('KMRY.png');
                    scale = 2.8959;
                else
                    airfield = imshow('KSNS.png');
                    scale = 2.7242;
                end

                button = 1;
                xy = []; n = 0;
                while n < 2
                    [xi,yi,button] = ginput(1);
                    n = n+1;
```

```
                    xy(:,n) = [xi;yi];
                end
                close('Select Start and End points.')

                len = scale*sqrt((xy(2,2)-xy(2,1))^2 + (xy(1,2)-
xy(1,1))^2);
                app.InspectionDistancemEditField.Value = len;
                app.UASEditField.Value =
UAS_Num(app,app.InspectionDistancemEditField.Value,app.Rwy_wd,...
                    app.UAS_alt,app.ModelDropDown.Value);

            end
            if app.FullSweepButton.Value == true
                app.InspectionDistancemEditField.Value = app.Rwy_len;
                close all;
            end
            app.DistanceScannedmEditField.Value = 1;
        end
        % Button pushed function: ResetDistanceScannedButton
        function ResetDistanceScannedButtonPushed(app, event)
            i = 0;
            app.DistanceScannedmEditField.Value = i;
            close all;
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [10 10 960 720];
            app.UIFigure.Name = 'UI Figure';
            app.UIFigure.Resize = 'off';
            % Create CoverageButtonGroup
            app.CoverageButtonGroup = uibuttongroup(app.UIFigure);
            app.CoverageButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @CoverageButtonGroupSelectionChanged, true);
            app.CoverageButtonGroup.Title = '3. Coverage';
            app.CoverageButtonGroup.Position = [512 306 196 103];
            % Create FullSweepButton
            app.FullSweepButton = uiradiobutton(app.CoverageButtonGroup);
            app.FullSweepButton.Text = 'Full Sweep';
            app.FullSweepButton.Position = [11 57 81 22];
            app.FullSweepButton.Value = true;
            % Create PartialSweepButton
            app.PartialSweepButton =
uiradiobutton(app.CoverageButtonGroup);
```

```
            app.PartialSweepButton.Text = 'Partial Sweep';
            app.PartialSweepButton.Position = [98 57 96 22];
            % Create InspectionDistancemEditFieldLabel
            app.InspectionDistancemEditFieldLabel =
uilabel(app.CoverageButtonGroup);
            app.InspectionDistancemEditFieldLabel.HorizontalAlignment =
'right';
            app.InspectionDistancemEditFieldLabel.Position = [34 33 131
22];
            app.InspectionDistancemEditFieldLabel.Text = 'Inspection
Distance (m)';
            % Create InspectionDistancemEditField
            app.InspectionDistancemEditField =
uieditfield(app.CoverageButtonGroup, 'numeric');
            app.InspectionDistancemEditField.Editable = 'off';
            app.InspectionDistancemEditField.Position = [43 12 113 22];
            % Create AirfieldMapPanel
            app.AirfieldMapPanel = uipanel(app.UIFigure);
            app.AirfieldMapPanel.Title = 'Airfield Map';
            app.AirfieldMapPanel.Position = [22 16 475 690];
            % Create WindEditFieldLabel
            app.WindEditFieldLabel = uilabel(app.AirfieldMapPanel);
            app.WindEditFieldLabel.HorizontalAlignment = 'right';
            app.WindEditFieldLabel.Position = [194 9 33 22];
            app.WindEditFieldLabel.Text = 'Wind';
            % Create WindEditField
            app.WindEditField = uieditfield(app.AirfieldMapPanel,
'numeric');
            app.WindEditField.Limits = [0 Inf];
            app.WindEditField.RoundFractionalValues = 'on';
            app.WindEditField.ValueDisplayFormat = '%.0f';
            app.WindEditField.Position = [242 9 40 22];
            % Create WindDirection
            app.WindDirection = uieditfield(app.AirfieldMapPanel, 'text');
            app.WindDirection.Position = [281 9 22 22];
            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            app.UIAxes.PlotBoxAspectRatio = [0.641425389755011 1
0.641425389755011];
            app.UIAxes.Box = 'on';
            app.UIAxes.Position = [32 56 451 621];
            % Create UASDisplayPanel
            app.UASDisplayPanel = uipanel(app.UIFigure);
            app.UASDisplayPanel.Title = '4. UAS Display';
            app.UASDisplayPanel.Position = [512 16 425 263];
            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.UASDisplayPanel);
            xlabel(app.UIAxes2, 'Relative Runway Width')
```

```
            ylabel(app.UIAxes2, 'Downrange')
            app.UIAxes2.PlotBoxAspectRatio = [1 0.513064133016627
0.513064133016627];
            app.UIAxes2.Position = [12 9 399 231];
            % Create DistanceScannedmEditFieldLabel
            app.DistanceScannedmEditFieldLabel = uilabel(app.UIFigure);
            app.DistanceScannedmEditFieldLabel.HorizontalAlignment =
'right';
            app.DistanceScannedmEditFieldLabel.Position = [713 332 124
22];
            app.DistanceScannedmEditFieldLabel.Text = 'Distance Scanned
(m)';
            % Create DistanceScannedmEditField
            app.DistanceScannedmEditField = uieditfield(app.UIFigure,
'numeric');
            app.DistanceScannedmEditField.Limits = [0 Inf];
            app.DistanceScannedmEditField.Editable = 'off';
            app.DistanceScannedmEditField.Position = [841 332 48 22];
            % Create InspectSwitchLabel
            app.InspectSwitchLabel = uilabel(app.UIFigure);
            app.InspectSwitchLabel.HorizontalAlignment = 'center';
            app.InspectSwitchLabel.Position = [893 288 44 22];
            app.InspectSwitchLabel.Text = 'Inspect';
            % Create InspectSwitch
            app.InspectSwitch = uiswitch(app.UIFigure, 'toggle');
            app.InspectSwitch.Items = {'Stop', 'Start'};
            app.InspectSwitch.ValueChangedFcn = createCallbackFcn(app,
@InspectSwitchValueChanged, true);
            app.InspectSwitch.Position = [905 335 18 42];
            app.InspectSwitch.Value = 'Stop';
            % Create CameraPanel
            app.CameraPanel = uipanel(app.UIFigure);
            app.CameraPanel.Title = '2. Camera';
            app.CameraPanel.Position = [512 415 425 142];
            % Create ModelDropDownLabel
            app.ModelDropDownLabel = uilabel(app.CameraPanel);
            app.ModelDropDownLabel.HorizontalAlignment = 'right';
            app.ModelDropDownLabel.Position = [6 93 38 22];
            app.ModelDropDownLabel.Text = 'Model';
            % Create ModelDropDown
            app.ModelDropDown = uidropdown(app.CameraPanel);
            app.ModelDropDown.Items = {'Zenmuse X3', 'Zenmuse Z3'};
            app.ModelDropDown.ValueChangedFcn = createCallbackFcn(app,
@ModelDropDownValueChanged, true);
            app.ModelDropDown.Position = [56 93 100 22];
            app.ModelDropDown.Value = 'Zenmuse X3';
            % Create FOVdegreesEditFieldLabel
            app.FOVdegreesEditFieldLabel = uilabel(app.CameraPanel);
```

```
            app.FOVdegreesEditFieldLabel.HorizontalAlignment = 'right';
            app.FOVdegreesEditFieldLabel.Position = [238 62 85 22];
            app.FOVdegreesEditFieldLabel.Text = 'FOV (degrees)';
            % Create FOVdegreesEditField
            app.FOVdegreesEditField = uieditfield(app.CameraPanel,
'numeric');
            app.FOVdegreesEditField.Editable = 'off';
            app.FOVdegreesEditField.Position = [336 62 44 22];
            % Create MinObjectSizemmEditFieldLabel
            app.MinObjectSizemmEditFieldLabel = uilabel(app.CameraPanel);
            app.MinObjectSizemmEditFieldLabel.HorizontalAlignment =
'right';
            app.MinObjectSizemmEditFieldLabel.Position = [202 32 121 22];
            app.MinObjectSizemmEditFieldLabel.Text = 'Min Object Size
(mm)';
            % Create MinObjectSizemmEditField
            app.MinObjectSizemmEditField = uieditfield(app.CameraPanel,
'numeric');
            app.MinObjectSizemmEditField.Editable = 'off';
            app.MinObjectSizemmEditField.Position = [336 32 44 22];
            % Create UASAltitudemSpinnerLabel
            app.UASAltitudemSpinnerLabel = uilabel(app.CameraPanel);
            app.UASAltitudemSpinnerLabel.HorizontalAlignment = 'right';
            app.UASAltitudemSpinnerLabel.Position = [6 62 95 23];
            app.UASAltitudemSpinnerLabel.Text = 'UAS Altitude (m)';
            % Create UASAltitudemSpinner
            app.UASAltitudemSpinner = uispinner(app.CameraPanel);
            app.UASAltitudemSpinner.Limits = [1 10];
            app.UASAltitudemSpinner.ValueChangedFcn =
createCallbackFcn(app, @UASAltitudemSpinnerValueChanged, true);
            app.UASAltitudemSpinner.Position = [111 62 55 22];
            app.UASAltitudemSpinner.Value = 1;
            % Create UASEditFieldLabel
            app.UASEditFieldLabel = uilabel(app.CameraPanel);
            app.UASEditFieldLabel.HorizontalAlignment = 'right';
            app.UASEditFieldLabel.Position = [8 32 40 22];
            app.UASEditFieldLabel.Text = '# UAS';
            % Create UASEditField
            app.UASEditField = uieditfield(app.CameraPanel, 'numeric');
            app.UASEditField.Limits = [0 Inf];
            app.UASEditField.Editable = 'off';
            app.UASEditField.Position = [129 32 36 22];
            % Create ImagesperLaneEditFieldLabel
            app.ImagesperLaneEditFieldLabel = uilabel(app.CameraPanel);
            app.ImagesperLaneEditFieldLabel.HorizontalAlignment = 'right';
            app.ImagesperLaneEditFieldLabel.Position = [6 5 106 22];
            app.ImagesperLaneEditFieldLabel.Text = '# Images per Lane';
            % Create ImagesperLaneEditField
```

```
            app.ImagesperLaneEditField = uieditfield(app.CameraPanel,
'numeric');
            app.ImagesperLaneEditField.Editable = 'off';
            app.ImagesperLaneEditField.Position = [129 5 37 22];
            % Create ResolutionEditFieldLabel
            app.ResolutionEditFieldLabel = uilabel(app.CameraPanel);
            app.ResolutionEditFieldLabel.HorizontalAlignment = 'right';
            app.ResolutionEditFieldLabel.Position = [217 93 62 22];
            app.ResolutionEditFieldLabel.Text = 'Resolution';
            % Create ResolutionEditField
            app.ResolutionEditField = uieditfield(app.CameraPanel,
'text');
            app.ResolutionEditField.Editable = 'off';
            app.ResolutionEditField.HorizontalAlignment = 'center';
            app.ResolutionEditField.Position = [296 93 84 22];
            app.ResolutionEditField.Value = '4000 x 3000';
            % Create InspectioninProgressLampLabel
            app.InspectioninProgressLampLabel = uilabel(app.UIFigure);
            app.InspectioninProgressLampLabel.HorizontalAlignment =
'right';
            app.InspectioninProgressLampLabel.Position = [722 366 124 22];
            app.InspectioninProgressLampLabel.Text = 'Inspection in
Progress';
            % Create InspectioninProgressLamp
            app.InspectioninProgressLamp = uilamp(app.UIFigure);
            app.InspectioninProgressLamp.Position = [861 366 20 20];
            app.InspectioninProgressLamp.Color = [1 0 0];
            % Create AirportRunwayPanel
            app.AirportRunwayPanel = uipanel(app.UIFigure);
            app.AirportRunwayPanel.Title = '1. Airport/Runway';
            app.AirportRunwayPanel.Position = [512 560 425 146];
            % Create AirportDropDownLabel
            app.AirportDropDownLabel = uilabel(app.AirportRunwayPanel);
            app.AirportDropDownLabel.HorizontalAlignment = 'right';
            app.AirportDropDownLabel.Position = [8 95 41 22];
            app.AirportDropDownLabel.Text = 'Airport';
            % Create AirportDropDown
            app.AirportDropDown = uidropdown(app.AirportRunwayPanel);
            app.AirportDropDown.Items = {'Singapore Changi Airport',
'Monterey Regional Airport', 'Salinas Municipal Airport'};
            app.AirportDropDown.ValueChangedFcn = createCallbackFcn(app,
@AirportDropDownValueChanged, true);
            app.AirportDropDown.Position = [64 95 188 22];
            app.AirportDropDown.Value = 'Singapore Changi Airport';
            % Create RunwayDropDownLabel
            app.RunwayDropDownLabel = uilabel(app.AirportRunwayPanel);
            app.RunwayDropDownLabel.HorizontalAlignment = 'right';
            app.RunwayDropDownLabel.Position = [8 72 49 22];
```

```
            app.RunwayDropDownLabel.Text = 'Runway';
            % Create RunwayDropDown
            app.RunwayDropDown = uidropdown(app.AirportRunwayPanel);
            app.RunwayDropDown.Items = {'02L/20R', '02C/20C'};
            app.RunwayDropDown.ValueChangedFcn = createCallbackFcn(app,
@RunwayDropDownValueChanged, true);
            app.RunwayDropDown.Position = [64 72 188 22];
            app.RunwayDropDown.Value = '02L/20R';
            % Create LengthmEditFieldLabel
            app.LengthmEditFieldLabel = uilabel(app.AirportRunwayPanel);
            app.LengthmEditFieldLabel.HorizontalAlignment = 'right';
            app.LengthmEditFieldLabel.Position = [8 48 63 22];
            app.LengthmEditFieldLabel.Text = 'Length (m)';
            % Create LengthmEditField
            app.LengthmEditField = uieditfield(app.AirportRunwayPanel,
'numeric');
            app.LengthmEditField.Editable = 'off';
            app.LengthmEditField.Position = [72 48 46 22];
            % Create WidthmEditFieldLabel
            app.WidthmEditFieldLabel = uilabel(app.AirportRunwayPanel);
            app.WidthmEditFieldLabel.HorizontalAlignment = 'right';
            app.WidthmEditFieldLabel.Position = [139 48 57 22];
            app.WidthmEditFieldLabel.Text = 'Width (m)';
            % Create WidthmEditField
            app.WidthmEditField = uieditfield(app.AirportRunwayPanel,
'numeric');
            app.WidthmEditField.Editable = 'off';
            app.WidthmEditField.Position = [202 48 46 22];
            % Create DateEditFieldLabel
            app.DateEditFieldLabel = uilabel(app.AirportRunwayPanel);
            app.DateEditFieldLabel.HorizontalAlignment = 'right';
            app.DateEditFieldLabel.Position = [266 95 31 22];
            app.DateEditFieldLabel.Text = 'Date';
            % Create DateEditField
            app.DateEditField = uieditfield(app.AirportRunwayPanel,
'text');
            app.DateEditField.Editable = 'off';
            app.DateEditField.Position = [312 95 104 22];
            % Create MaxSpeedEditFieldLabel
            app.MaxSpeedEditFieldLabel = uilabel(app.AirportRunwayPanel);
            app.MaxSpeedEditFieldLabel.HorizontalAlignment = 'right';
            app.MaxSpeedEditFieldLabel.Position = [288 72 66 22];
            app.MaxSpeedEditFieldLabel.Text = 'Max Speed';
            % Create MaxSpeedEditField
            app.MaxSpeedEditField = uieditfield(app.AirportRunwayPanel,
'numeric');
            app.MaxSpeedEditField.Editable = 'off';
            app.MaxSpeedEditField.Position = [369 72 47 22];
```

```
            % Create UASSpeedEditFieldLabel
            app.UASSpeedEditFieldLabel = uilabel(app.AirportRunwayPanel);
            app.UASSpeedEditFieldLabel.HorizontalAlignment = 'right';
            app.UASSpeedEditFieldLabel.Position = [288 50 68 22];
            app.UASSpeedEditFieldLabel.Text = 'UAS Speed';
            % Create UASSpeedEditField
            app.UASSpeedEditField = uieditfield(app.AirportRunwayPanel,
'numeric');
            app.UASSpeedEditField.Editable = 'off';
            app.UASSpeedEditField.Position = [369 49 47 22];
            % Create SpeedmsSliderLabel
            app.SpeedmsSliderLabel = uilabel(app.AirportRunwayPanel);
            app.SpeedmsSliderLabel.HorizontalAlignment = 'right';
            app.SpeedmsSliderLabel.Position = [225 29 71 22];
            app.SpeedmsSliderLabel.Text = 'Speed (m/s)';
            % Create SpeedmsSlider
            app.SpeedmsSlider = uislider(app.AirportRunwayPanel);
            app.SpeedmsSlider.Limits = [1 22];
            app.SpeedmsSlider.ValueChangedFcn = createCallbackFcn(app,
@SpeedmsSliderValueChanged, true);
            app.SpeedmsSlider.ValueChangingFcn = createCallbackFcn(app,
@SpeedmsSliderValueChanging, true);
            app.SpeedmsSlider.Position = [311 38 104 3];
            app.SpeedmsSlider.Value = 1;
            % Create ResetDistanceScannedButton
            app.ResetDistanceScannedButton = uibutton(app.UIFigure,
'push');
            app.ResetDistanceScannedButton.ButtonPushedFcn =
createCallbackFcn(app, @ResetDistanceScannedButtonPushed, true);
            app.ResetDistanceScannedButton.Position = [729 297 148 22];
            app.ResetDistanceScannedButton.Text = 'Reset Distance
Scanned';
        end
    end
    methods (Access = public)
        % Construct app
        function app = AFDS2
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
```

```
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

APPENDIX B. MATLAB CODE FOR OBJECT DETECTION

```
clc
close all
clear all
tic
%%
%naming conventions - input files by UAV height
%output files in hxx0y, where h is UAV hgt, xx is filter size,
%y where 1 is redbox image, 2 is output image

FolderName = 'W3x3'; %Folder where images with suspected FOD are stored

cd 'D:\School\Working Matlab\MML Baseline'
%read base files, organized from 1m to 10m
originalfiles = dir('*.jpg');
nfiles = length(originalfiles);    % Number of files found
for ii=1:nfiles
   originalfilename = originalfiles(ii).name;
   originalimage = imread(originalfilename);
   oldimages{ii} = originalimage;
end

%read new images, organized from 1m to 10m
WorkingFolder = strcat('D:\School\Working Matlab\',FolderName);
cd(WorkingFolder);
imagefiles = dir('*.jpg');
nfiles = length(imagefiles);    % Number of files found
for ii=1:nfiles
   currentfilename = imagefiles(ii).name;
   currentimage = imread(currentfilename);
   images{ii} = currentimage;
   gps{ii}=imfinfo(currentfilename);
end

%%
for ii=1:nfiles
    figure(ii*10000);
    pair{ii} = ii*10000;
    filename = sprintf('%i.png', pair{ii});
    %filter images after converting to gray
    fixed = rgb2gray(oldimages{ii});
    moving = rgb2gray(images{ii});
    [ht, wd, dim] = size(images{ii});

    [optimizer, metric] = imregconfig('multimodal');
    optimizer.InitialRadius = 0.003;
    optimizer.Epsilon = 1.5e-7;
    optimizer.GrowthFactor = 1.005;
```

```
    optimizer.MaximumIterations = 900;

    %tform and align image
    tform = imregtform(moving, fixed, 'affine', optimizer, metric);
    movingRegistered =
imwarp(moving,tform,'OutputView',imref2d(size(fixed)));
    imshowpair(fixed, movingRegistered,'Scaling','joint')
    set(gca,'position',[0 0 1 1],'units','normalized')
    gf = getframe(gcf);
    hd = fullfile('D:','School','Working Matlab','Results',FolderName,...
        'ImgPair',filename);
    imwrite(gf.cdata,hd);

    %find difference between images
    %g refers to filter size
    for f=1:12
        g=f*2+1;
        redbox{g} = ii*10000+g*100+1;
        redFileName = sprintf('%i.png', redbox{g});
        output{g} = ii*10000+g*100+2;
        outFileName = sprintf('%i.png', output{g});
        diffname{g} = ii*10000+g*100+3;
        diffFileName = sprintf('%i.png', diffname{g});
        befbox{g} = ii*10000+g*100+4;
        befName = sprintf('%i.png', befbox{g});
        diff=imabsdiff(fixed,movingRegistered);
        figure(ii*10000+g*100+3);
        imshow(diff)
        set(gca,'position',[0 0 1 1],'units','normalized')
        difr=getframe(gcf);
        dn = fullfile('D:','School','Working
Matlab','Results',FolderName,...
            'Diff_bef_filt',diffFileName);
        imwrite(difr.cdata,dn);
        close;
        highlighted_diff = medfilt2(roicolor(diff,100,200),[g g]);

        %draw bounding box around suspected FODs
        figure(ii*10000+g*100+1);
        imshow(highlighted_diff);
        set(gca,'position',[0 0 1 1],'units','normalized')
        befr=getframe(gcf);
        befboxn = fullfile('D:','School','Working
Matlab','Results',FolderName,...
            'Before_box',befName);
        imwrite(befr.cdata,befboxn);
        st=regionprops(highlighted_diff, 'BoundingBox');
        for k = 1 : length(st)
            thisBB = st(k).BoundingBox;
            rectangle('Position',
[thisBB(1),thisBB(2),thisBB(3),thisBB(4)],...
```

```
            'EdgeColor','r','LineWidth',2)
        end

        %save image for further processing
        fr=getframe(gcf);
        imwrite(fr.cdata,'save.png');
        fn = fullfile('D:','School','Working
Matlab','Results',FolderName,...
            'Redbox',redFileName);
        imwrite(fr.cdata,fn);
        img=imresize(imread('save.png'),[ht wd]);

        %separate channels to filter out the red boxes
        red = img(:,:,1); % Red channel
        green = img(:,:,2); % Green channel
        blue = img(:,:,3); % Blue channel
        boxes=imsubtract(imsubtract(red,green),blue);

        %save and display output images
        figure(ii*10000+g*100+2);
        imshow(boxes);
        %disp(g)
        imshowpair(movingRegistered,boxes,'diff');
        set(gca,'position',[0 0 1 1],'units','normalized')
        kr=getframe(gcf);
        kn = fullfile('D:','School','Working
Matlab','Results',FolderName,...
            'Output',outFileName);
        imwrite(kr.cdata,kn);
        delete save.png
        close;
        close;
    end
end

% obtain and store the processing time for this folder
watch = toc
watch_min = toc/60
timefile = strcat('D:\School\Working Matlab\Results\',FolderName);
textfile = fopen(strcat(timefile,'\time.txt'),'a+');
fprintf(textfile,'Total Elapsed Time:%.4g seconds / %.4g minutes\n',...
    watch,watch_min);

cd 'D:\School\Working Matlab'
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Alexander-Adams, Marcia. 2013. “Fact Sheet - Foreign Object Debris (FOD).” Fact Sheet. Washington, DC: Federal Aviation Authority. https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=15394.

Chambers, Joshua, and Rohaidi, Nurfilzah. 2017. “How Can Singapore Address Its Manpower Shortage?” *GovInsider* (blog), May 25, 2017. https://govinsider.asia/digital-gov/how-can-singapore-address-its-manpower-shortage/.

Chew, Khien Meow David. 2011. Runway surveillance system and method. U.S. Patent 9,483,952, filed March 17, 2011, and issued November 1, 2016. http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=1&f=G&l=50&co1=AND&d=PTXT&s1=stratech.ASNM.&s2=chew.INNM.&OS=AN/stratech+AND+IN/chew&RS=AN/stratech+AND+IN/chew.

Department of the Air Force. 2015. *Airfield Operations Procedures and Programs.* Air Force Instruction 13-204, Volume 3. http://static.e-publishing.af.mil/production/1/af_a3/publication/afi13-204v3/afi13-204v3.pdf.

Federal Aviation Authority. 2009. *Airport Foreign Object Debris (FOD) Detection Equipment.* Advisory Circular 150/5220-24. Washington, DC: Federal Aviation Authority.

Google Maps. 2018. “McMillan Airfield / East Perimeter.” Accessed July 6, 2018. https://www.google.com/maps/place/McMillan+Airfield+%2F+East+Perimeter/@35.7174078,-120.7720719,4202m/data=!3m1!1e3!4m5!3m4!1s0x80ecd3ca0619c4f5: 0x8ee6d45bd8f4929d!8m2!3d35.715083!4d-120.763032.

Harney, Robert. C. 2004. “Combat Systems. Volume 1— Sensor Elements. Part 1 - Sensor Functional Characteristics.” cnqzu.com. http://cnqzu.com/library/Anarchy%20Folder/Electronics%20and%20Communications/Electronic%20Warfare/Combat%20System%20Sensors.pdf.

Hogan, Sean D., Maggi Kelly, Brandon Stark, and YangQuan Chen. 2017. “Unmanned Aerial Systems for Agriculture and Natural Resources.” *California Agriculture* 71 (1): 8-14. http://calag.ucanr.edu/archive/?type=pdf&article=ca. 2017a0002.

Hubbard, Sarah, Andrew Pak, Yue Gu, and Yan Jin. 2017. “UAS to Support Airport Safety and Operations: Opportunities and Challenges,” *Journal of Unmanned Vehicle Systems*. September 11 2017. https://doi.org/10.1139/juvs-2016-0020.

Jain, Ramesh, Rangachar Kasturi, and Brian G. Schunck. 1995. *Machine Vision.* New York: McGraw-Hill.

Leite, Clélio. 2014. *SENTINEL System Foreign Object Debris (FOD) Detection Performance Evaluation Trial - June.* Report Number TR-01P-2014. Willis, TX: Pharovision. http://pharovision.com/documents/Sentinel%20FOD%20Trials%20Report.pdf.

McCreary, Iain. 2010. *Runway Safety: FOD, Birds, and the Case for Automated Scanning.* Washington, DC: Insight SRI.

National Population and Talent Division. 2013. *A Sustainable Population for a Dynamic Singapore: Population White Paper*. Singapore: Ministry of Trade and Industry.

Ong, Justin. 2018. "Singapore's Fertility Rate at New 7-Year Low of 1.16: Josephine Teo." Channel News Asia, March 01, 2018. https://www.channelnewsasia.com/news/singapore/singapore-total-fertility-rate-new-low-1-16-10002558.

Pedersen, Christian. S. 2017. "Singapore's Productivity Challenge." *The Straits Times.* June 9, 2017. https://www.straitstimes.com/opinion/singapores-productivity-challenge.

PIX4D. 2018. "Computing the Flight Height for a Given GSD." Accessed July 11, 2018. https://support.pix4d.com/hc/en-us/articles/202557469.

PR Newswire. 2017. "Robird And Integrated Drone Solutions Deployed At Major International Airport." May 9, 2017. https://www.prnewswire.com/news-releases/robird-and-integrated-drone-solutions-deployed-at-major-international-airport-300453338.html.

Seow, Joanna. 2017. "Singapore must manage inflow of new immigrants carefully: PM Lee Hsien Loong." *Straits Times.* May 28, 2017. https://www.straitstimes.com/singapore/spore-must-manage-inflow-of-new-immigrants-carefully-pm.

Sherry, Lance. 2009. "Introduction to Airports Design and Operations." Class notes for SYST460/560-Fall 2009: Introduction to Airports Design and Operations, George Mason University, Center for Air Transportation Systems Research, Fairfax, VA. http://catsr.ite.gmu.edu/SYST460/IntroAirportsWorkbook.pdf.

Wang, Ching-Wei, and Hsiang-Chou Chen. 2013. "Improved Image Alignment Method in Application to X-ray Images and Biological Images." *Bioinformatics.* 29, (15): 1879-1887. https://doi.org/10.1093/bioinformatics/btt309.

Whitehead, Sam. 2017. "How an Atlanta Airport Is Using Drones to Help with Runway Maintenance." Marketplace. May 31, 2017. https://www.marketplace.org/2017/05/31/business/how-atlanta-airport-using-drones-help-with-runway-maintenance.

Wile, Rob. 2017. “Why High-End Drones Are Half the Price They Were a Year Ago.” *Time,* June 2 2017. http://time.com/money/4800984/drone-prices-decrease-spark-dji/.

Worland, Justin. 2017. “The FAA Allowed Drones to Fly at an Airport for the First Time Ever.” *Fortune.* February 2, 2017. http://fortune.com/2017/02/02/faa-drones-airport-atlanta-hartsfield-jackson/.

Xie, Yaowen, Linlin Li, Haoyu Wang, and Xiaojiong Zhao. 2011. “Spatio-temporal processes and causes analysis of Jiayuguan Oasis in China over a 23a period,” *19th International Conference Geoinformatics*, pp. 1-4, 2011. https://doi.org/10.1109/GEOINFORMATICS.2011.5980679.

Yakimenko, Oleg. 2011. *Engineering Computations and Modeling in MATLAB®/Simulink®.* VA: AIAA.

Yakimenko, Oleg. 2018. “Video Scoring for Automated Bursts Detection.” Unpublished paper, 20 May 2018.

Yuan, Chi, Zhixiang Liu, and Youmin Zhang. 2015. “UAV-based Forest Fire Detection and Tracking Using Image Processing Techniques.” In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*: 639-643. https://doi.org/10.1109/ICUAS.2015.7152345.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California