NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

EXPEDITIONARY LOGISTICS: A LOW-COST, DEPLOYABLE, UNMANNED AERIAL SYSTEM FOR AIRFIELD DAMAGE ASSESSMENT

by

Nicholas J. Davis

December 2018

Thesis Advisor:	Gurminder Singh
Second Readers:	Charles D. Prince
	Michael R. McCarrin

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

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

4. TITLE AND SUBTITLE EXPEDITIONARY LOGISTICS: A LOW-COST, DEPLOYABLE, UNMANNED AERIAL SYSTEM FOR AIRFIELD DAMAGE ASSESSMENT	**5. FUNDING NUMBERS**
6. AUTHOR(S) Nicholas J. Davis	
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)

Airfield Damage Repair (ADR) is among the most important expeditionary activities for our military. The goal of ADR is to restore a damaged airfield to operational status as quickly as possible. Before the process of ADR can begin, however, the damage to the airfield needs to be assessed. As a result, Airfield Damage Assessment (ADA) has received considerable attention. Often in a damaged airfield, there is an expectation of unexploded ordnance, which makes ADA a slow, difficult, and dangerous process. For this reason, it is best to make ADA completely unmanned and automated. Additionally, ADA needs to be executed as quickly as possible so that ADR can begin and the airfield restored to a usable condition. Among other modalities, tower-based monitoring and remote sensor systems are often used for ADA. There is now an opportunity to investigate the use of commercial-off-the-shelf, low-cost, automated sensor systems for automatic damage detection. By developing a combination of ground-based and Unmanned Aerial Vehicle sensor systems, we demonstrate the completion of ADA in a safe, efficient, and cost-effective manner.

14. SUBJECT TERMS airfield damage assessment, airfield damage repair, 3D point cloud, image stitching, neural network, object detection, unmanned Aerial vehicle, unmanned aerial system	**15. NUMBER OF PAGES** 103
	16. PRICE CODE

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

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

THIS PAGE INTENTIONALLY LEFT BLANK

EXPEDITIONARY LOGISTICS: A LOW-COST, DEPLOYABLE, UNMANNED AERIAL SYSTEM FOR AIRFIELD DAMAGE ASSESSMENT

Nicholas J. Davis
Lieutenant, United States Navy
BS, Air Force Academy, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2018

Approved by: Gurminder Singh
Advisor

Charles D. Prince
Second Reader

Michael R. McCarrin
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Airfield Damage Repair (ADR) is among the most important expeditionary activities for our military. The goal of ADR is to restore a damaged airfield to operational status as quickly as possible. Before the process of ADR can begin, however, the damage to the airfield needs to be assessed. As a result, Airfield Damage Assessment (ADA) has received considerable attention. Often in a damaged airfield, there is an expectation of unexploded ordnance, which makes ADA a slow, difficult, and dangerous process. For this reason, it is best to make ADA completely unmanned and automated. Additionally, ADA needs to be executed as quickly as possible so that ADR can begin and the airfield restored to a usable condition. Among other modalities, tower-based monitoring and remote sensor systems are often used for ADA. There is now an opportunity to investigate the use of commercial-off-the-shelf, low-cost, automated sensor systems for automatic damage detection. By developing a combination of ground-based and Unmanned Aerial Vehicle sensor systems, we demonstrate the completion of ADA in a safe, efficient, and cost-effective manner.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
- A. CONTEXT AND MOTIVATION 1
- B. PROBLEM STATEMENT 2
- C. RESEARCH QUESTIONS 2
- D. PROJECT SCOPE 2
- E. BENEFIT OF STUDY 3
- F. THESIS OUTLINE 3

II. BACKGROUND INFORMATION 5
- A. AIRFIELD DAMAGE REPAIR 5
- B. AIRFIELD DAMAGE ASSESSMENT 5
  - 1. Airfield Damage Assessment Phase 1—Initial Reconnaissance 6
  - 2. Airfield Damage Assessment Phase 2 7
- C. EMERGENCY OPERATIONS CENTER 8
- D. POTENTIAL ISSUES WITH CURRENT PROCESSES 10
- E. TRADITIONAL AND EXPEDITIONARY AIRFIELD DAMAGE REPAIR 11
- F. RELEVANT COMPUTER TECHNOLOGY FOR ADA 11
  - 1. Image Stitching and Mosaic Creation 11
  - 2. Orthophoto and Photogrammetry 12
  - 3. TensorFlow 12
  - 4. Unmanned Aerial Systems 13
- G. RELATED WORK 14
  - 1. iFerret Hydra 14
  - 2. Geospatial Expeditionary Planning Tool 15
  - 3. Concrete Building Inspection 16
  - 4. Road Inspection 17
  - 5. Large Area Survey 18
  - 6. Object Detection using Neural Networks 19
- H. SUMMARY 20

III. IMPLEMENTATION AND EXPERIMENT DESIGN 21
- A. SYSTEM GOAL 21
- B. SYSTEM ARCHITECTURE OVERVIEW 21
- C. GENERAL SYSTEM SET UP 23
- D. INITIAL RUNWAY SCAN 24
  - 1. UAV Flight over the Runway 24

2. Data Transfer 26
3. Damage Detection 28
4. Damage Localization 34
5. Mosaic Creation 36
E. SECOND RUNWAY SCAN 38
F. REPAIR PLANNING 41
G. CHAPTER SUMMARY 41

IV. IMPLEMENTATION AND TESTING 43
A. UAV DATA FLOW 43
1. Live Video Stream 43
2. Flight Controller Preference 43
3. Caching of Image and Video Data 44
4. Wi-Fi and SD Card Data Transfer 44
B. WEBODM AND AGISOFT PHOTOSCAN 45
C. CPU AND GPU TENSORFLOW 45
D. POTHOLE DETECTION VS TWO-CLASS DAMAGE DETECTION 46
E. GPU TENSORFLOW AND IMAGE STITCH CONFLICT 47
F. UAV PICTURE TRIGGER MODE DURING IMAGE CAPTURE 48
G. RUNWAY IMAGE REQUIREMENTS AND IMAGE STITCHING SPEED 49
H. GRID LINES ERROR DUE TO IMAGE ROTATION 52
I. DETECTION OPTIMIZATION AND PERFORMANCE EVALUATION 52
1. Initial Test 53
2. Generalized Damage Detection 57
3. Data Set Optimization 60
4. Generalized Damage Detection with New Data Set 63
5. Comparison of Results 66
J. CHAPTER SUMMARY 69

V. CONCLUSION AND FUTURE WORK 71
A. SUMMARY 71
B. CONCLUSION 72
C. FUTURE WORK 72

APPENDIX. VIRTUAL ENVIRONMENT SET UP 75

LIST OF REFERENCES 77

INITIAL DISTRIBUTION LIST 81

LIST OF FIGURES

Figure 1. Examples of spalls and craters. Source: DAF (2012)................................6

Figure 2. Information guide for reporting damage during ADA Phase 2. Source: DAF (2012)..8

Figure 3. Mock up example of an airfield damage plot. Source: DAF (2012). 9

Figure 4. Full picture plot in order to determine MOS. Source: DAF (2012)...........10

Figure 5. Phantom 4 Pro made by DJI. Source: Phantom 4 (2018). 14

Figure 6. Static position iFerret camera overlooking airfield. Source: iFerret (2015)..15

Figure 7. Example view of GeoExPT visual layout of an airfield. Source: GeoExPT (2018)..16

Figure 8. Drone used to inspect concrete cracks. Source: Kim et al. (2017). 17

Figure 9. UAV imagery of a rural road and corresponding 3D model. Source: Zhang and Elaksher (2010)...18

Figure 10. UAV mapped area with plotted error size at each location. Source: Siebert and Teizer (2014)...19

Figure 11. Examples of detecting pedestrians using a trained neural network. Source: Angelova et al. (2015). 20

Figure 12. Overview of current airfield repair process. Source: DAF (2012). 22

Figure 13. System architecture high level design..22

Figure 14. Design overview of initial runway scan architecture................................24

Figure 15. Pix4D user interface. Source: Pix4D (2018)..26

Figure 16. Live video data feed system setup. 27

Figure 17. Examples of varying pothole images from ImageNet Source: Pothole Synset (2010). 29

Figure 18. Sliding window technique..30

Figure 19. Interface for LabelImg program..31

Figure 20. Original image input into the system. 33

Figure 21. Example output of analyzed image. 33

Figure 22. Overview of damage localization system design. 34

Figure 23. Visualization of GSD parameters. Source: What is ground sample distance (GSD) and how does it affect your drone data? (2018). 35

Figure 24. Equations to calculate GSD. Source: What is ground sample distance (GSD) and how does it affect your drone data? (2018). 35

Figure 25. Example mosaic of a runway produced from a series of images taken from a UAV. 37

Figure 26. Example mosaic of a runway produced from a series of images taken from a UAV with automatic damage detection. 38

Figure 27. Example mosaic of a runway on ramp produced from a series of images taken from a UAV with grid lines overlaid every 10 meters based on GPS data. 38

Figure 28. 2D and 3D grid options in Pix4D for mission planning 39

Figure 29. Example output of a 3D model produced with Agisoft PhotoScan. 40

Figure 30. Single class damage detection on airfield damage caused by heavy machinery. 46

Figure 31. Single class damage detector model on airfield damage caused by explosives. 47

Figure 32. Mosaic showing vanishing effect due to UAV flight parameter settings. 49

Figure 33. Mosaic created over a distance of 270.61 meters using 7 equally spaced images. 50

Figure 34. Mosaic created over a distance of 270.61 meters using 3 equally spaced images. 51

Figure 35. Example of differing detection results of the same image rotated on all four axes. 54

Figure 36. Example of grouping multi-detection. 55

Figure 37. Examples of images from data set that lead to poor detection results. 61

Figure 38. Comparison of Precision between two-classification and damage detection mode with original and modified data sets.................................66

Figure 39. Comparison of Recall between two-classification and damage detection model with original and modified data sets. 67

Figure 40. Comparison of Accuracy between two-classification and damage detection model with original and modified data sets. 67

Figure 41. Comparison of Negative Predictive Rate between two-classification and damage detection model with original and modified data sets. 68

Figure 42. Comparison of Negative Predictive Rate between two-classification and damage detection model with original and modified data sets. 68

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Relationship between ground distance, number of images, image stitch processing time and mosaic quality. 50

Table 2. Relationship between varying ground distance, number of images, image stitch processing time, and mosaic quality.....................................51

Table 3. Results of two-class model with 5% error threshold in width and height...55

Table 4. Results of two-class model with 10% error threshold in width and height...56

Table 5. Results of two-class model with 15% error threshold in width and height...56

Table 6. Results of two-class model with 20% error threshold in width and height...57

Table 7. Results of using two-class model to conduct damage detection without classification with 5% error threshold in width and height. 58

Table 8. Results of using two-class model to conduct damage detection without classification with 10% error threshold in width and height. 58

Table 9. Results of using two-class model to conduct damage detection without classification with 15% error threshold in width and height. 59

Table 10. Results of using two-class model to conduct damage detection without classification with 20% error threshold in width and height. 59

Table 11. Results of two-class model with 5% error threshold in width and height with modified data set. 61

Table 12. Results of two-class model with 10% error threshold in width and height with modified data set. 62

Table 13. Results of two-class model with 15% error threshold in width and height with modified data set. 62

Table 14. Results of two-class model with 20% error threshold in width and height with modified data set. 63

Table 15. Results of using two-class model to conduct damage detection without classification with 5% error threshold in width and height with modified data set. 64

Table 16. Results of using two-class model to conduct damage detection without classification with 10% error threshold in width and height with modified data set. 64

Table 17. Results of using two-class model to conduct damage detection without classification with 15% error threshold in width and height with modified data set. 65

Table 18. Results of using two-class model to conduct damage detection without classification with 20% error threshold in width and height with modified data set. 65

LIST OF ACRONYMS AND ABBREVIATIONS

ADA	airfield damage assessment
ADR	airfield damage repair
AFPAM	Air Force pamphlet
CAD	Computer Aided Design
COTS	commercial off the shelf
CPU	central processing unit
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
CUPTI	CUDA Profiling Tools Interface
CV	computer vision
DoD	Department of Defense
EOC	Emergency Operations Center
FOD	Foreign Object Damage
GCP	ground control point
GeoExPT	Geospatial Expeditionary Planning Tool
GPS	global positioning system
GPU	graphics processing unit
GSD	ground sample distance
GUI	graphical user interface
MAOS	minimum airfield operating strip
MOS	minimum operating strip
ODM	Open Drone Map
RTS	robotic total station
TTP	tactics, techniques, and procedures
UAS	unmanned aerial system
UAV	unmanned aerial vehicle
UXO	unexploded ordnance

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The ability to conduct air operations is critical to the success of all the branches of the Department of Defense (DoD). Air operations are not limited to air-to-air combat; they also support ground and sea operations. Air power includes a variety of roles and responsibilities including air interdiction, intelligence collection, battlefield management, etc. In addition, the use of air power is not limited to times of war. Heavy use of aircraft occurs outside of war environments in crucial tasks such as moving personnel and equipment during peace keeping, humanitarian assistance, disaster relief, and training operations. Regardless of the objective, all aircraft have a basic need for a functional airfield. In a war time environment, not only is this need exceptionally increased but so is the likelihood that enemy forces will damage our airfields.

Current Navy Airfield Damage Assessment (ADA) and Airfield Damage Repair (ADR) processes are dangerous, slow, iterative, inaccurate, and human-in-the-loop dependent. The emergence of cheap, readily available, and accurate technology offers a way to increase the efficiency and accuracy of the ADA process, assist the repair decision making process, and help expedite airfield readiness for take-off and landing operations.

The goal of this thesis is to develop and test a system architecture and define the data flow to conduct a fast and accurate ADA using a combination of commercial-off-the-shelf (COTS) sensors and equipment. Our system uses a camera-equipped UAV combined with multiple image analysis and processing techniques to assist the Emergency Operations Center (EOC) in minimum operating strip (MOS) determination and planning of repair operations. Our method is comprised of two distinct components: initial and secondary runway scans.

The initial runway scan consists of multiple sub components, including the UAV flight over the damaged runway, automated damage detection, damage localization and mosaic creation. The goal of the initial runway scan is to collect and forward the most critical information to the EOC as fast as possible in order to quickly make a determination if the runway is damaged, worth repairing, assist in MOS selection, and visualize, identify,

and classify damage to the maximum extent possible. We accomplish this by using survey mapping techniques built into Pix4Dcapture, a Phantom 4 Pro UAV and an iPad/flight controller, to take a series of overlapping images across the runway. We then use a trained TensorFlow neural network to automatically identify and classify the damage. This information is both logged for direct output and also rendered visually by projecting results back onto the image itself. Finally, the images are stitched together and grid lines are placed onto the resulting mosaic using Python's OpenCV library.

The second runway scan takes place after the initial runway scan is finished and a MOS is selected. Once the area that requires repairs has been identified, the UAV can be sent back to previously logged damaged locations, and use survey mapping techniques in Pix4Dcapture in order to collect more images of the area of interest. These images can then be used to create 3D models in programs such as WebODM and Agisoft PhotoScan. Finally, the 3D models can be incorporated into existing modeling software such as GeoExPT/AutoCAD in order to estimate required repair materials.

Our system offers a cost-effective and low-training-requirement design that is capable of assisting the EOC in conducting ADA. We identified a set of tools and software, tested them, and developed a data flow that produces a mosaic of an airfield, overlays grid lines onto the mosaic, automatically identifies and logs the position of spalls and craters, and generates 3D models for further analysis. These products can have an immediate and direct positive impact on the EOC's ability to conduct ADA by increasing the accuracy and decreasing the time to conduct the assessment. In addition, our system lays the groundwork for continued development and integration with multiple sensors and pieces of technology to create an even more capable solution for automated ADA.

ACKNOWLEDGMENTS

I would like to thank Dr. Gurminder Singh, Mr. Carl Prince, and Mr. Michael McCarrin for their continuous support and guidance throughout this research project. I would also like to thank the members of the Air Force Civil Engineering Center for their assistance in data and information sharing. Additionally, a big thanks goes to Rithvik Chuppala, for his work on a custom drone interface application, and to Angela Hu, for her work in labeling an airfield damage dataset. Finally, I'd like to say thank you to my wife, Alexandra, for her continuous love and support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter introduces the Airfield Damage Assessment (ADA) problem set and discusses the scope and goals of this research project. It also outlines the format and describes the contents of each chapter.

A. CONTEXT AND MOTIVATION

The ability to conduct air operations is critical to the success of all the branches of the Department of Defense (DoD) is. Air operations are not limited to air-to-air combat; they also support ground and sea operations. Air power includes a variety of roles and responsibilities including air interdiction, intelligence collection, battlefield management, etc. In addition, the use of air power is not limited to times of war. Heavy use of aircraft occurs outside of war environments in crucial tasks such as moving personnel and equipment during peace keeping, humanitarian assistance, disaster relief, and training operations. Regardless of the objective, all aircraft have a basic need for a functional airfield. In a war time environment, not only is this need exceptionally increased but so is the likelihood that enemy forces will damage our airfields. Without the assistance of technology, the process of conducting an accurate assessment of a damaged airfield is a slow, labor intensive, and dangerous process.

Current baseline for Navy airfield repair is based on Air Force Tactics, Techniques, and Procedures (TTPs). Although there is considerable overlap with the Air Force Mission, the Navy problem has a slightly different scope. From a functional standpoint the problem of having to repair an airfield is the same. However, the Navy mission assumes that we will be expeditionary, forward operating, and repairing an airfield that may or may not have previously been controlled by U.S. or friendly forces. Consequently, the assumption must be made that there are little to no pre-positioned resources so all personnel, equipment, and supplies will need to be either air dropped or brought it by an already space limited ship. In addition, it is likely that these repair operations will be in a hostile environment so speed and accuracy are paramount to the ADA problem.

B. PROBLEM STATEMENT

Current Navy ADA and Airfield Damage Repair (ADR) processes are dangerous, slow, iterative, inaccurate, and human-in-the-loop dependent. The emergence of cheap, readily available, and accurate technology offers a way to possibly increase the efficiency of and/or automate some parts of the ADA/ADR process in order to decrease the required time and increase the accuracy of a damage assessment, assist the repair decision making process, and expedite airfield repair.

C. RESEARCH QUESTIONS

- How can a time and resource effective system be developed using commercial-off-the-shelf (COTS) components to decrease the completion time and increase the accuracy of our current ADA process?
- With what speed, precision, recall and accuracy can we detect, localize, classify, and map/report airfield damage?

D. PROJECT SCOPE

The focus of this project is to evaluate the speed and accuracy of conducting an ADA using a COTS unmanned aerial vehicle (UAV) and to develop a method for providing information required by the Emergency Operations Center (EOC) in order to determine minimum operating strip (MOS) and plan repair operations. This project focuses on the data flow and system architecture in order to achieve the above and lays the foundation for follow-on system enhancements. The goals of this project are as follows:

- Display entire runway with accurate grid overlaid
- Automatically detect/log of damage on runway
- Produce a system that is faster than current manual processes

This research does not evaluate the cost effectiveness of different drones or test effectiveness of different sensors. Further, we do not attempt to detect debris or unexploded ordnance (UXO). We leave the task of establishing secure communication between various parts of the system architecture to future work. Finally, we make no attempt to detect that an attack has taken place or to calculate volume estimates from generated 3D models.

E. BENEFIT OF STUDY

This study provides a proof of concept system for improving our ability to conduct ADA. In addition, this project will lay the foundation for a more accurate, robust, multi-sensor solution. Finally, this project aims to identify a methodology and data flow in order to minimize processing time and maximize result accuracy.

F. THESIS OUTLINE

Chapter II defines ADA, ADR, and discusses current processes used to conduct these operations. It also introduces various technologies that may be applied to this problem set and related work in this field.

Chapter III discusses system design and overall project methodology. In addition, it describes data flow and implementation for each part of the project.

Chapter IV explains the testing of the system including system metrics, findings, performance and accuracy. It also covers the system limitations and possible enhancements.

Chapter V concludes with the findings of this study and lists out possible areas for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND INFORMATION

This chapter discusses the importance of ADA, describes the state-of-the-art and current process of conducting ADA/ADR, introduces technology that may be used to enhance and expedite these operations, and discusses related research in the field of ADA.

A. AIRFIELD DAMAGE REPAIR

ADR operations consists of actions taken in order to return an airfield and associated infrastructure back to operating condition as quickly as possible (Department of the Air Force [DAF], 2012). These operations may be necessary, both at friendly airfields that have been damaged, and at enemy airfields that friendly forces have taken over. Air Force Pamphlet (AFPAM) titled "Airfield Damage Repair Operations", describes the ADR process in three separate phases (DAF, 2012). The first phase, after security for the airfield has been established, is to "Open the Runway." This involves activities such as removing large obstacles from the runway, Unexploded Ordnance (UXO) identification and removal, and inspecting the runway for various types of damages. This phase is where ADA is conducted. The second phase, once the airfield has been repaired, is establishing critical lighting and airfield markings. The final phase are the actions necessary to employ sustainment troops to maintain the airfield and switch from hasty to normal operations.

B. AIRFIELD DAMAGE ASSESSMENT

ADA is a critical component that takes place during phase one of the overarching ADR operation. ADA includes actions taken to evaluate the damage that most directly impacts airfield operations to include damage on runways and taxiways (DAF, 2012). The main goal of the ADA is to locate, identify, and classify UXO and various types of damage on the airfield including spalls, craters, and camouflets so that repair operations can begin. Examples of this type of damage are illustrated in Figure 1. The ADA process itself is broken down into two separate phases: Initial Reconnaissance and Detailed Reconnaissance (DAF, 2012).

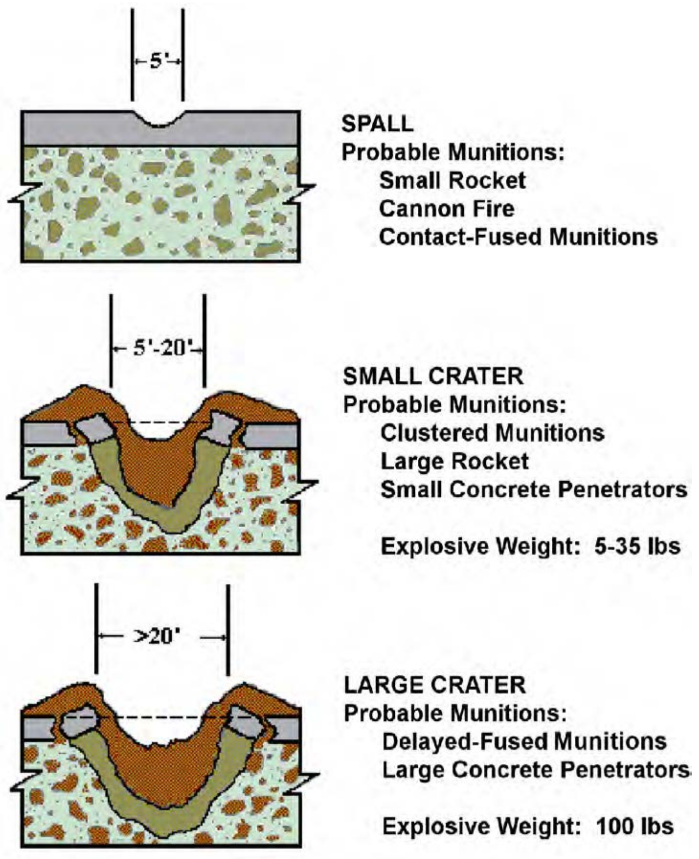


Figure 1. Examples of spalls and craters. Source: DAF (2012).

1. Airfield Damage Assessment Phase 1—Initial Reconnaissance

The goal of ADA Phase 1 is to detect air field damage as quickly as possible at the expense of accuracy (DAF, 2012). Units from various observation posts including airfield towers, fixed cameras, or pre-determined lookout locations simply observe as much as they can from their current position and report the information. Accuracy is welcome but not a requirement for this phase. A "quick look" is a safe and expedited way to aggregate visual information which may be enough to determine the extent, general location and various types of damage that have taken place, if any at all. If a more detailed look at the airfield is required in order to ascertain the extent and location of the damage, ADA Phase 2 must be conducted.

2. Airfield Damage Assessment Phase 2

If further information on the amount of airfield damage is required, Phase 2, "Detailed Reconnaissance" will commence which will be much more dangerous and take much longer to complete compared to Phase 1 (DAF, 2102). The goal of ADA Phase 2 is to locate and classify all damage and UXO. The process can either be done on foot or, at the exchange of decreased accuracy for increased safety, in armored vehicles (DAF, 2012). If on foot, approximately 6–8 troops gather at one end of the airfield, line up at double-arm interval in order to spread the width of the airfield, then slowly walk down the length of the airfield documenting all types and sizes of damage and UXO. The locations and sizes are still very approximate, usually based on distance markers on the side of the airfield, but is much more accurate than estimates produced in Phase 1. As they proceed down the airfield, one person follows behind the line of airfield damage assessors in order to record all the information. Periodically, the recorder will relay segments of the information to the Emergency Operations Center (EOC), the role of which is described in the following section. A reporting checklist with mandatory requirements for each piece of damage can be seen in Figure 2. These assessors will continue their assessment until they have walked the expanse of the entire airfield

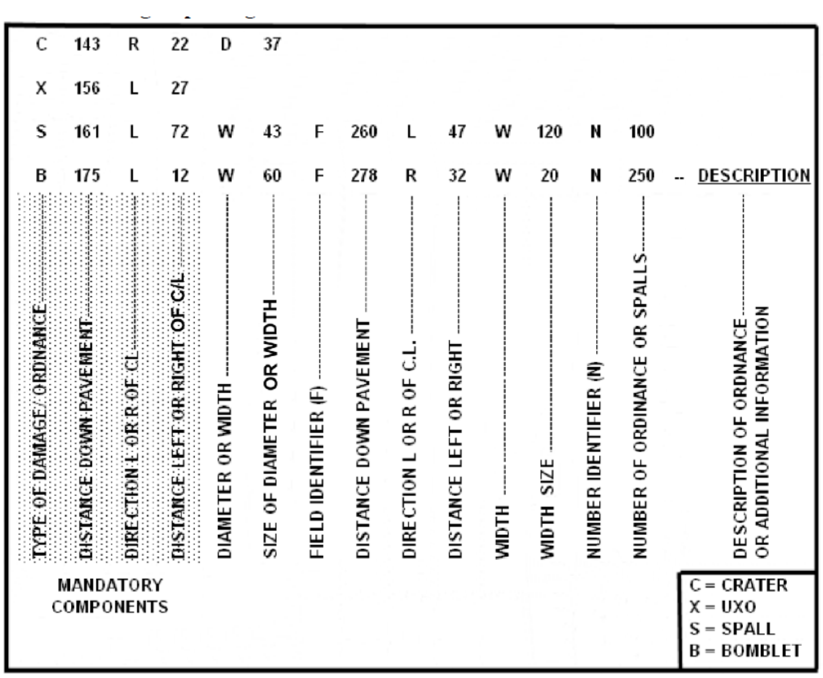


Figure 2. Information guide for reporting damage during ADA Phase 2. Source: DAF (2012).

C. EMERGENCY OPERATIONS CENTER

The EOC is main operations center and the focal point for all active ADR operations (DAF, 2012). During ADR operations, leadership, key decision makers, and ADR experts are located at the EOC, which could be near the airstrip or possibly miles away depending on the operational environment. While ADA units are conducting their assessment and relaying information, units at the EOC aggregate that information onto a paper mock-up of the entire airfield. An example can be seen in Figure 3. Once ADA units have completed their assessment, units at the EOC determine the Minimum Operating Strip (MOS) and begin planning on how to conduct ADR based on the information received from ADA units, including UXO removal, UXO safety stand-off, estimated materials required, equipment needed, etc. The MOS is comprised of only the areas required for an aircraft to take off and land (DAF, 2012). This is determined by the extent of the damage on the airfield and what type of aircraft need to be able to land. Heavier aircraft have different requirements than smaller aircraft in both runway length and impact strength. An example of an entire air base with plotted damage can be seen in Figure 4. Repair operations must wait for MOS to be selected in order for maximum efficiency in the repair process. Once the MOS has been selected and repair operations are underway, units at the EOC also plan for repair of the Minimum Airfield Operating Strip (MAOS) which consists of not only the runway but also supporting elements such as aircraft taxiways (DAF, 2012).

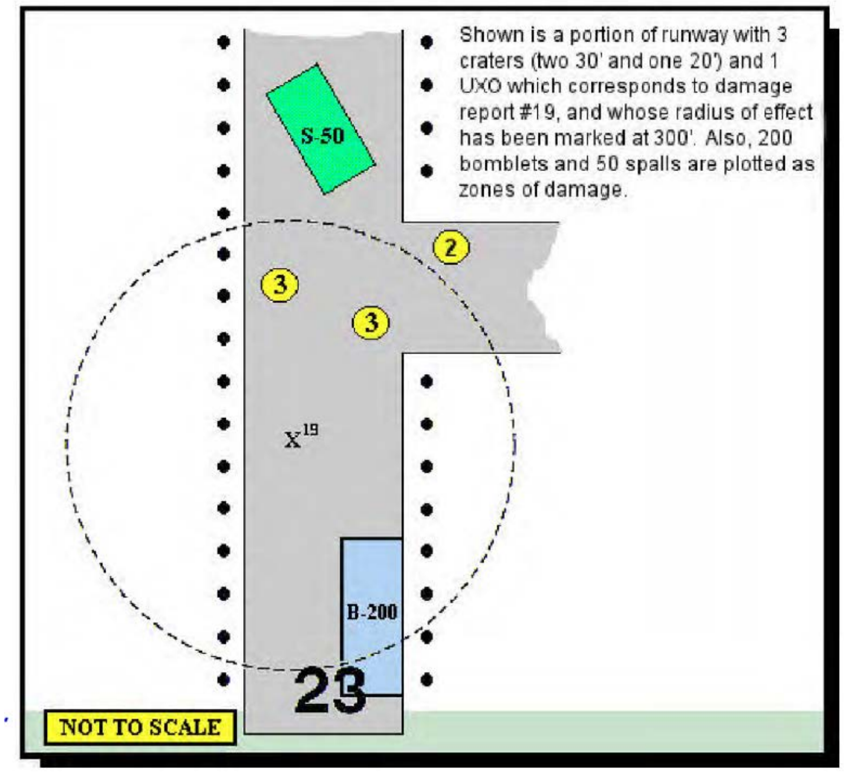

Shown is a portion of runway with 3 craters (two 30' and one 20') and 1 UXO which corresponds to damage report #19, and whose radius of effect has been marked at 300'. Also, 200 bomblets and 50 spalls are plotted as zones of damage.
S-50
2
3
3
X19
B-200
23
NOT TO SCALE

Figure 3. Mock up example of an airfield damage plot.
Source: DAF (2012).

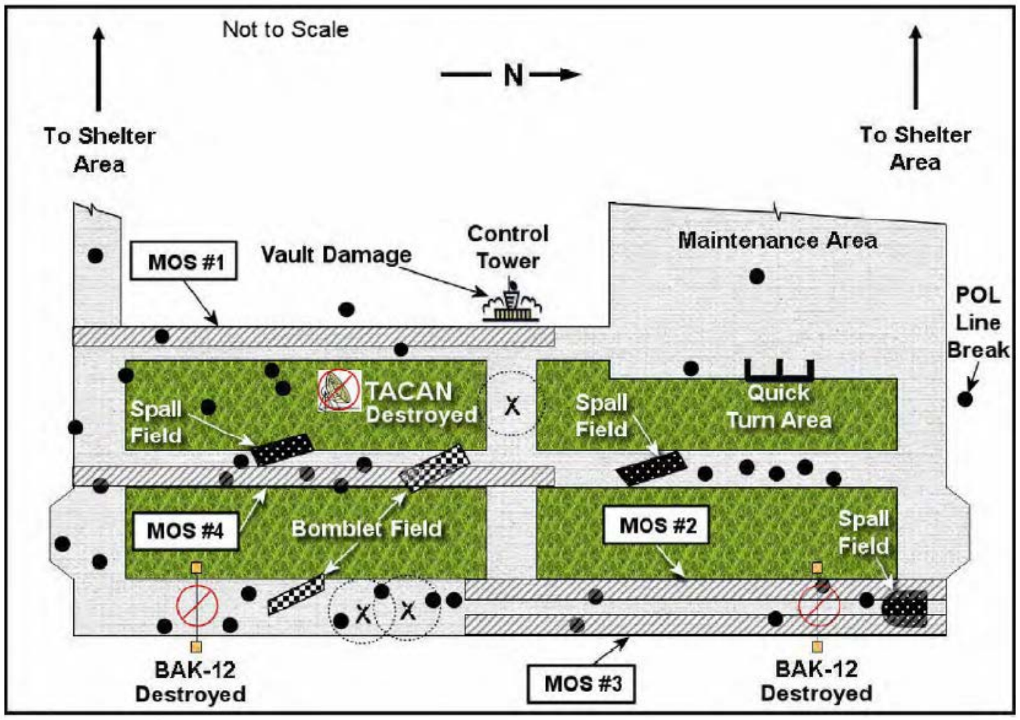


Figure 4. Full picture plot in order to determine MOS. Source: DAF (2012).

D. POTENTIAL ISSUES WITH CURRENT PROCESSES

The ADA process in support of ADR operations is functional but has areas that could be improved upon. One potential issue is the possibility of misinformation given the multiple times that someone has to verbally pass the same information regarding types and location of damage. At a minimum, whoever identifies the damage during the ADA phase verbally passes the information to a recorder, who then transmit the data to the EOC where it is recorded and then translated onto the mock-up airfield. However, the EOC is not necessarily co-located with the airfield and may be multiple radio hops away introducing more potential points for error. Another issue is the accuracy of the data. Due to safety stand-off, utilization of pre-placed runway markers, and a hand worked airfield map, estimates of scale and location of damage can be inaccurate. These inaccuracies can potentially lead to non-optimal MOS selection or a gross overestimation or underestimation of required resource materials to repair the airfield. Finally, time is a critical factor in this process. The faster an accurate assessment can be made, the faster repair operations can begin and the faster air operation can begin. The amount of damage that needs to be recorded and the safety standoff zones from UXOs can make the process of evaluating a multi kilometer airfield very time consuming.

E. TRADITIONAL AND EXPEDITIONARY AIRFIELD DAMAGE REPAIR

It is worth noting the difference in process between traditional and expeditionary ADA and ADR operations. In traditional ADA/ADR where the airfield is already owned and controlled by friendly forces, it is likely that support elements including repair materials and equipment are already present. This improves tolerance for reduced accuracy in ADA reports because material repair requirements can be overestimated since repair material is likely not only readily available but also available en masse. In addition, in traditional ADA/ADR, units should already have a very detailed understanding of the airfield layout leading to a much quicker ability to determine MOS. In contrast to this is expeditionary ADA/ADR, where the Navy will likely find itself operating. There is likely little to no functional on-sight repair material, equipment or support infrastructure. Consequently, expeditionary ADA/ADR puts a higher importance on correct material estimates since everything will have to be brought in. In addition, accurate location data during the ADA process is also very important since available information about the airfield may be limited (DAF, 2012). Finally, due to the expeditionary nature of Navy ADA and subsequent lack of resources, it is likely the data shown in Figure 3 and Figure 4 will simply be drawn up manually either with pen and paper or on a white board further adding to potential areas where human error may occur.

F. RELEVANT COMPUTER TECHNOLOGY FOR ADA

This section introduces various technology that can be used to increase the speed and/or accuracy of conducting an ADA in support of ADR operations.

1. Image Stitching and Mosaic Creation

Szelinski in “Image Alignment and Stitching: A Tutorial,” describes multiple image alignment and feature matching techniques that can be leveraged to create accurate mosaics. In a series of images that have some overlap, similar features can be identified in each image either through pixel intensity or feature matching that can then be used as anchors to accurately blend multiple images together (Szeliski, 2005). The percentage overlap between images is a critical factor for image stitching in a time constrained environment. More overlap yields more potential features for matching, likely resulting in a more accurate image stitch however, larger overlap also requires more images to cover the same area requiring more computation time for both data transmission and processing time.

2. Orthophoto and Photogrammetry

An orthophoto is an image that has been corrected in order to be an accurate representation of the earth's surface including ground features (Smith, n.d.). Photogrammetry is the ability to make determinations about objects without directly interacting with those objects (Schenk, 2005). Combining an orthophoto with photogrammetry techniques allows the ability to measure true distance between points or objects in an image. This is especially useful when trying to determine how large an object is or how deep a hole is in a particular 2D image.

3. TensorFlow

TensorFlow is an open source machine learning library. Machine learning encompasses the idea of using algorithms to learn from a set of data in order to make predictions in a particular environment (Copeland, 2016). An artificial neural network is a machine learning framework inspired by biological neural networks. A neural network attempts to mimic the same way humans learn and has numerous real-world applications such as image and speech recognition and natural language processing (Nielsen, 2015). For the best results, it is critical that the data on which the neural network is trained be as representative as possible to the data that it will eventually be used on (Hulstaert, 2018).

In systems that use machine learning to detect objects in images, there is often a tradeoff between speed and accuracy. For example, one approach is to perform object recognition on a small sliding window that moves across the image repeatedly inspecting its contents for objects that the neural network was trained on. However, although this method would likely deliver the most accurate results, it would be extremely slow and computationally expensive. At the expense of some accuracy, speed could be improved by enlarging the window or minimizing the window overlap. Further developments include clustering groups of pixels based on differentiation from their neighbors and then conducting the object detection on each cluster. This type of system attempts to minimize the number of windows that need to be inspected in order to increase speed at the expense of accuracy. An object detection system does not have to use a single technique or rule set. Many models use a combination of techniques in order to try and maximize accuracy and efficiency (Hulstaert, 2018).

4. Unmanned Aerial Systems

The rapid advancement of UAVs offers a cheap, user friendly, and effective way for users to collect data using custom built payloads over large distances. A clear front runner in the consumer drone world is DJI. The Phantom 4 Pro is a COTS UAV designed and sold by DJI and can be seen in Figure 5. It has a maximum flight time of roughly 30 minutes, a maximum range of about 7 km, GPS enabled, an integrated 20MP camera, and an assortment of built in sensors such as infrared for automated collision avoidance and aerial maneuvers. This drone can be flown from DJI apps such as DJI GO 4 but also has an open API for custom flight app development. In addition, apps to control the drone can be used on both IOS and Android systems. Having an open architecture, custom software, full access to numerous sensors, flexibility of operating system, and the ability to add custom payloads results in a cost efficient and effective way for users to collect data over large distances very quickly (Phantom 4, 2018).

Figure 5. Phantom 4 Pro made by DJI. Source: Phantom 4 (2018).

G. RELATED WORK

This section introduces current technologies and research that conduct, assist, or may be used to enhance ADA operations.

1. iFerret Hydra

iFerret Hydra is a system that can automatically detect Foreign Object and Debris (FOD) on a runway. In addition, the system has numerous other features such as the capability to conduct measurements on FOD. Being a video-based system allows the information to be broadcasted in near-real-time (iFerret, 2015). The system is comprised of a series of fixed multi-sensor stations that are statically placed along airfields and runways. An example can be seen in Figure 6. In addition to typical HD electro optical sensors, the system also utilizes electro-optic sensors and millimeter wave radar in order to conduct operations at night or in adverse weather conditions (iFerret Hydra, 2015). This system can be further integrated with Stratech's Super BullsEye II technology in order to detect and measure weapon damage on land and on water (Super BullsEye II, 2015). Having fixed and elevated structures along runways that may be attacked presents a possible issue as it is impractical to use this system to assess an airfield that does not already have it in place or one that was severely damaged in an attack. However, Stratech overcomes some of these shortfalls by incorporating their technology onto a vehicle in order to achieve mobile FOD detection and damage assessment (iFerret Mobile, 2015).

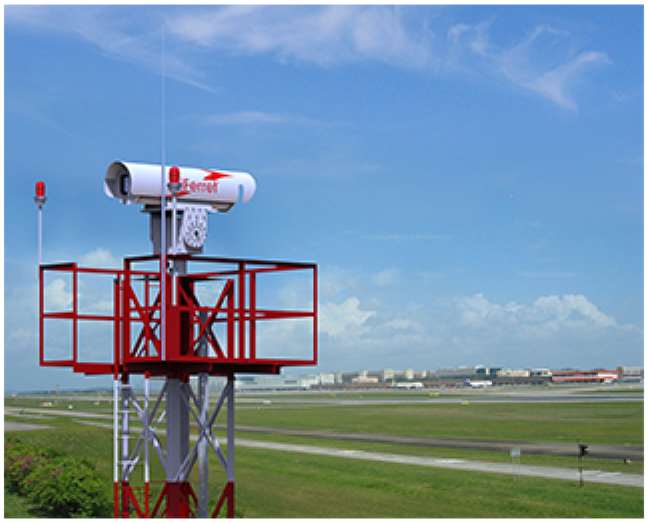
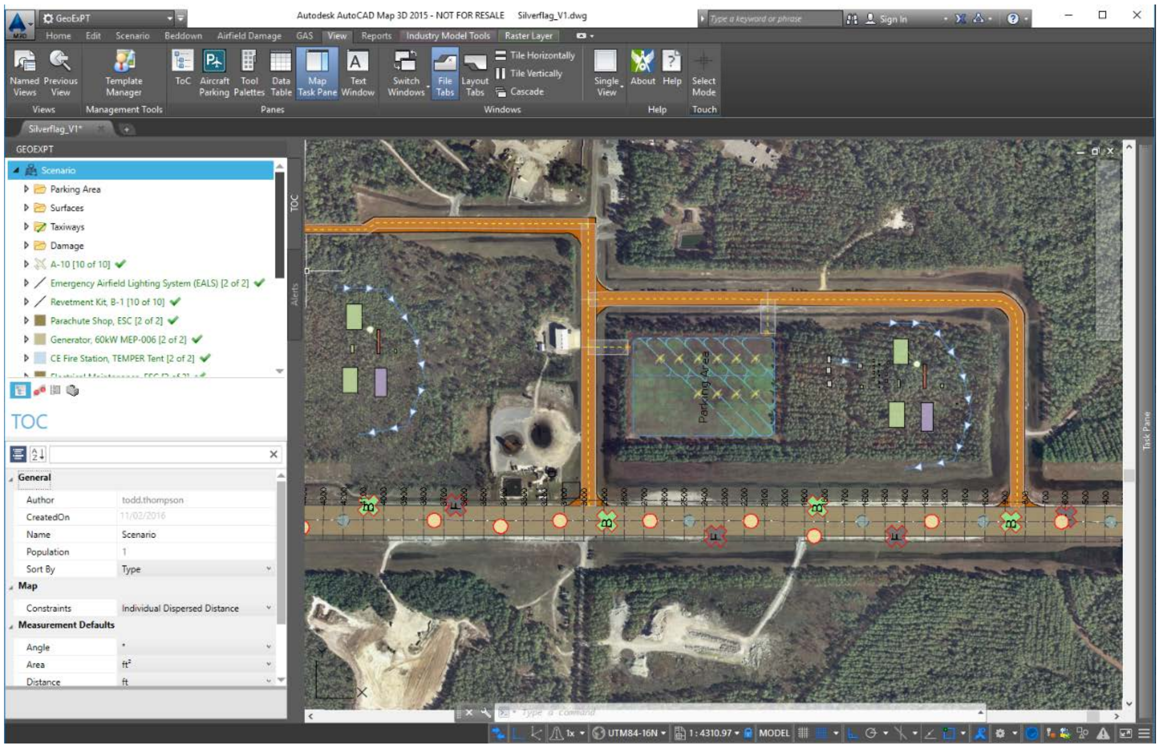


Figure 6. Static position iFerret camera overlooking airfield.
Source: iFerret (2015).

2. Geospatial Expeditionary Planning Tool

Geospatial Expeditionary Planning Tool (GeoExPT) is an ADA support tool that is used to manually create a digital to-scale version of an airfield in order to plot structures, aircraft, and damage. Once an airfield has been created, the tool can be used to plot various types of airfield damage and provides various tools in order to determine Minimum Airfield Operating Surface (MOAS). GeoExPT is built as a plug in for Autodesk AutoCAD Map 3D so it also has access to all standard AutoCAD tools and capabilities as well (GeoExPT, 2018). GeoExPT is currently in use across the DoD in order to plan and execute ADA and ADR operations. An example of an airfield layout using GeoExPT can be seen in Figure 7.


Autodesk AutoCAD Map 3D 2015 - NOT FOR RESALE Silverflag_V1.dwg
Home Edit Scenario Beddown Airfield Damage GAS View Reports Industry Model Tools Raster Layer
Named Views Previous View Template Manager ToC Aircraft Parking Tool Palettes Data Table Map Task Pane Text Window Switch Windows File Tabs Layout Tabs Tile Horizontally Tile Vertically Cascade Single View About Help Select Mode
Views Management Tools Panes Windows Help Touch
Parking Area
TOC

GEOEXPT

- Scenario
  - Parking Area
  - Surfaces
  - Taxiways
  - Damage
  - A-10 [10 of 10] ✓
  - Emergency Airfield Lighting System (EALS) [2 of 2] ✓
  - Revetment Kit, B-1 [10 of 10] ✓
  - Parachute Shop, ESC [2 of 2] ✓
  - Generator, 60kW MEP-006 [2 of 2] ✓
  - CE Fire Station, TEMPER Tent [2 of 2] ✓

General	
Author	todd.thompson
CreatedOn	11/02/2018
Name	Scenario
Population	1
Sort By	Type
Map	
Constraints	Individual Dispersed Distance
Measurement Defaults	
Angle	°
Area	ft²
Distance	ft

Figure 7. Example view of GeoExPT visual layout of an airfield. Source: GeoExPT (2018).

3. Concrete Building Inspection

Utilizing UAV capabilities is already an established practice for conducting surveys in potentially dangerous areas. In “Concrete Crack Identification Using a UAV Incorporating Hybrid Image Processing” (Kim, Lee, Ahn, Cho, Shin & Sim, 2017), researchers presented a strategy utilizing UAVs to monitor cracks in concrete structures, a critical component of assessing structure damage and safety. Their system works by post-processing data collected with a combination of sensors. They found that they were able to accurately measure cracks as long as they were thicker than .1 mm with a maximum length estimation error of 7.3% (Kim et al., 2017). Figure 8 shows their custom UAV set up for their experimentation. More generally, their research shows that UAVs offer accessibility in that they can accurately inspect areas that are not directly accessible to people or may be simply too dangerous to inspect manually. In addition to accessibility, UAVs provide the capability to carry a custom payload of sensors and the communications equipment required to transmit the data of interest.

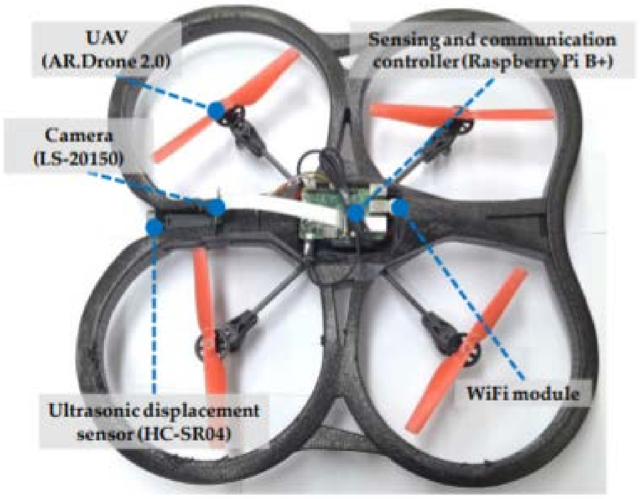


Figure 8. Drone used to inspect concrete cracks.
Source: Kim et al. (2017).

4. Road Inspection

Chunsun Zhang and Ahmed Elaksher showed another possible application of UAV imagery in “3D Reconstruction from UAV-acquired Imagery for Road Surface Distress Assessment” (Zhang & Elaksher, 2010). By using a UAV with a camera, they created 3D models of roads from 2D images by using a model that employed several image matching algorithms at different levels of detail. If accurate enough, these 3D models could then be used in order to measure distress of rural roads. Their research concludes that the techniques explored are a viable way to automate the examination of rural roads for the benefit of the U.S. Department of Transportation (Zhang & Elaksher, 2010). Figure 9 shows an example from their work of a transformation from a 2D image of a road to a 3D rendering that can be used to calculate distress.

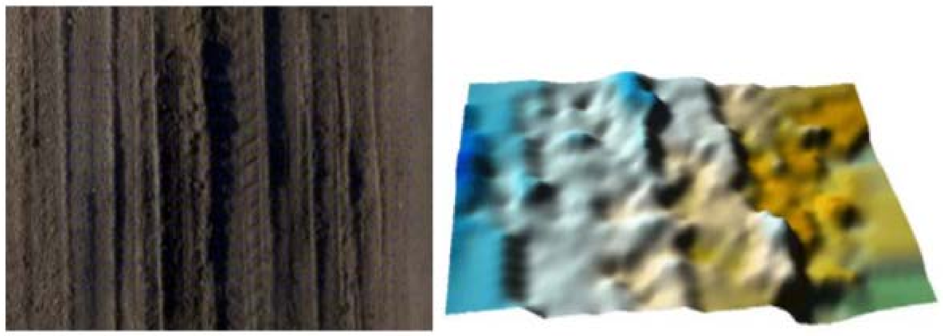


Figure 9. UAV imagery of a rural road and corresponding 3D model. Source: Zhang and Elaksher (2010).

5. Large Area Survey

Siebert and Dr. Teizer, in their article titled “Mobile 3D Mapping for Surveying Earthwork Using an Unmanned Aerial Vehicle (UAV),” show how a UAV can be used to quickly and effectively gather information over large areas in order to construct 3D models (Siebert & Teizer, 2014). They successfully demonstrate the capability of a program that, when given certain parameters by the user, can generate a flight path for a UAV in order to automate the surveying process. In addition, they conduct a survey over a large area and compare the results from the geo-referenced UAV data against the traditional Robotic Total Station (RTS) measurements (Siebert & Teizer, 2014). Figure 10 shows a plot of the area that they surveyed using a UAV during their study. Their research concludes that UAVs can be used to conduct a large area survey. However, there are some current physical limitations specific to UAVs that may need to be addressed with such a long flight time including battery life and weather/environmental restrictions.

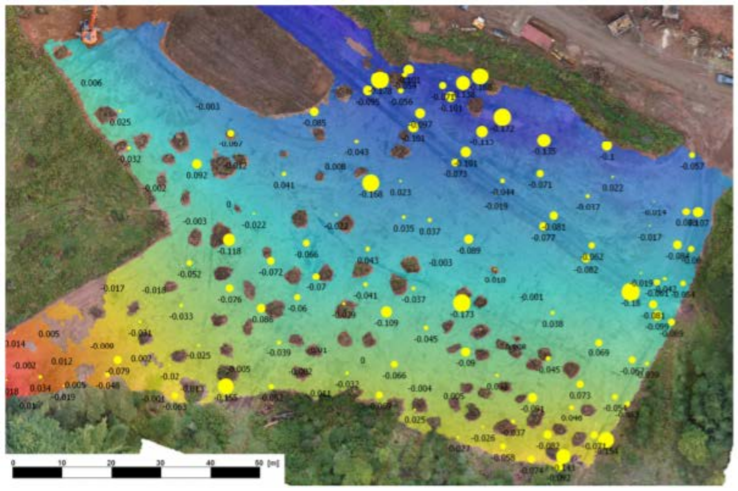


Figure 10. UAV mapped area with plotted error size at each location. Source: Siebert and Teizer (2014).

6. Object Detection using Neural Networks

Neural networks must not only be accurate but must also be fast and efficient. This is especially important when dealing with time sensitive applications such as vehicle safety. In “Pedestrian Detection with a Large-Field-Of-View Deep Network,” Angelova, Krizhevsky, and Vanhoucke demonstrate a workflow for automatically detecting pedestrians (Angelova, Krizhevsky, Vanhoucke, 2015). Instead of a traditional sliding window technique which can require a large number of positions in an image to be processed in order to achieve high accuracy, they used a large-field-of-view deep neural network to minimize the number of positions required to be processed in each image in an effort to decrease processing time (Angelova et al., 2015). With a slight expense to recall and no loss in precision, they were able to speed up the detection process by more than 60 times.

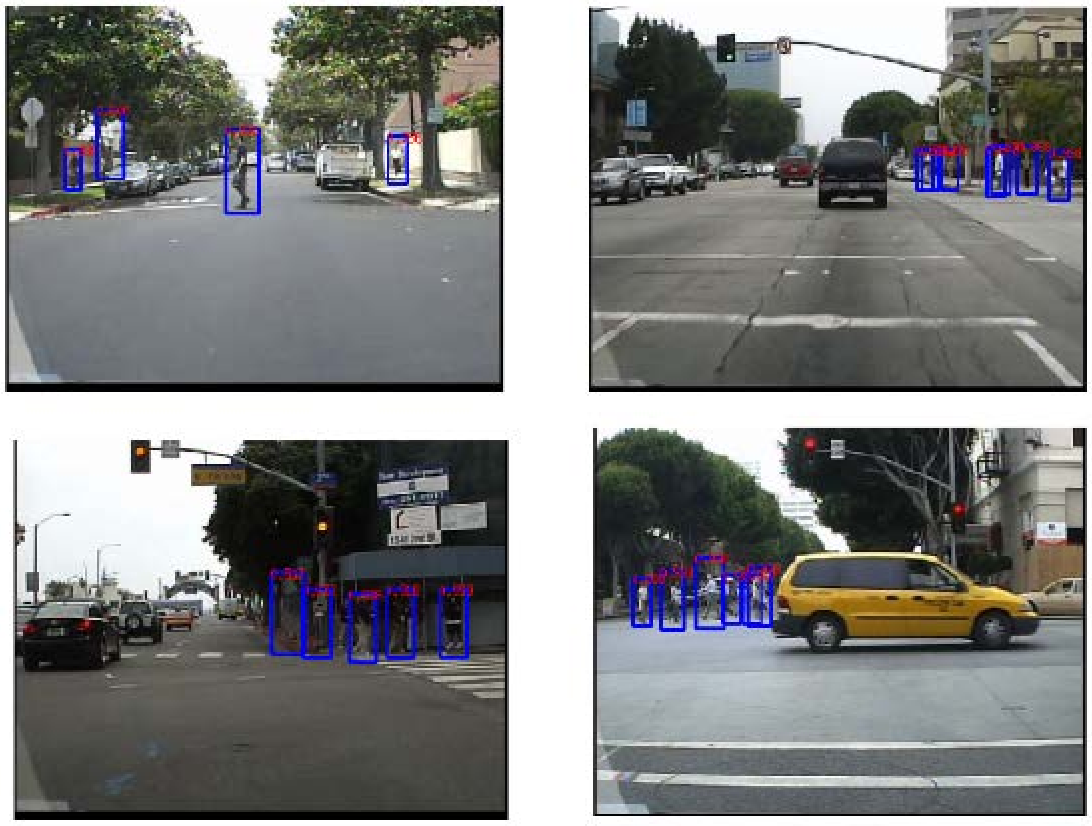


Figure 11. Examples of detecting pedestrians using a trained neural network. Source: Angelova et al. (2015).

H. SUMMARY

This chapter has provided an introduction to ADA and how it relates to the greater ADR architecture. We described the phases that comprise the assessment process and, although this process is effective, present some of the reasons why it needs to be improved upon. Finally, we give a survey of related work and the various computer techniques that can be applied to the ADA/ADR problem set. Chapter III explains our methodology and defines a proposed system architecture that combines various technologies in order to enhance the speed, efficiency, and accuracy of ADA operations.

III. IMPLEMENTATION AND EXPERIMENT DESIGN

This chapter discusses the data flow, algorithms, and software applications used in order to develop our ADA system. First, the overall goal of the system is discussed followed by an overview of the entire system process and how it fits in current ADA procedures. Then follow on explanations of each component of the overall system are provided including general system architecture, survey mapping process, object detection training and utilization, and image stitching/mosaic creation. Lastly, the use of 3D modeling software in order for better visualization and conducting complex measurements is discussed.

A. SYSTEM GOAL

This system is designed to fulfill the critical and time sensitive data gathering needs of the EOC in order to determine MOS. Specifically, these needs include an image mosaic of the airfield, damage detection, and damage localization. Once the necessary data has been processed and MOS has been selected, the system is designed to facilitate follow on actions including more detailed data gathering in areas of interest in order to create 3D models and assist in the estimation of repair materials required.

B. SYSTEM ARCHITECTURE OVERVIEW

This section describes the high-level overview of the implementation of ADA for this project. This system is designed to be employed during the Airbase Assessment/Opening portion of the greater ADA process shown in Figure 12. Figure 13 further defines the process and information flow of this project.

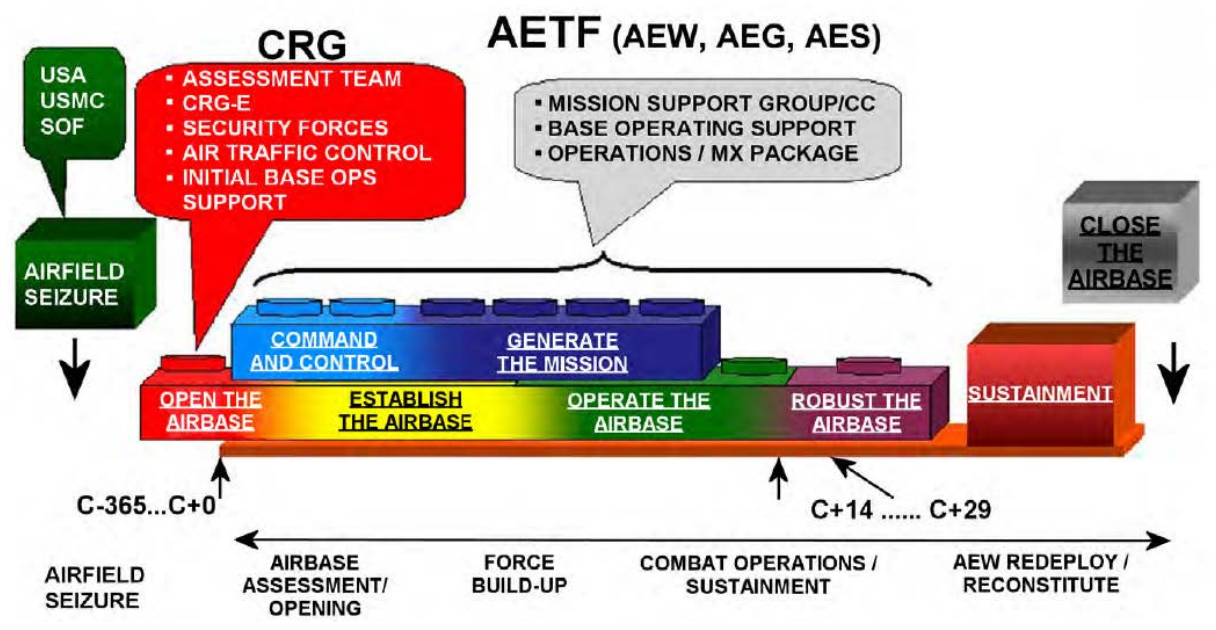


Figure 12. Overview of current airfield repair process. Source: DAF (2012).

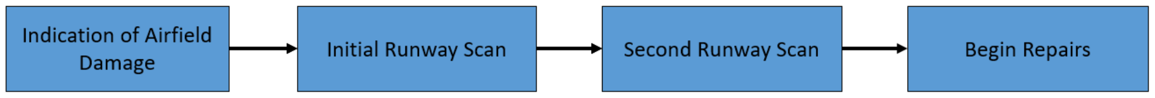


Figure 13. System architecture high level design.

This system is designed to be run after an attack has taken place and is prompted by some notification that an assessment needs to be conducted. It is also designed to be image source agnostic. Parameters can be set within the system in order to correct measurements for the specific kind of camera and UAV used.

The Initial Runway Scan is the first step in the system process and begins after receiving a notification that an assessment needs to be conducted. This scan is conducted in order to provide critical high-level information to the EOC as quickly as possible in order to determine MOS. Fine grained details can be overlooked at this point in an effort to increase speed and efficiency. This step ends after the UAV has conducted its first scan, the data has been processed and an MOS has been selected.

The second runway scan begins after the initial runway scan and processing of the data has been completed. After an MOS is selected, the UAV conducts a second scan to collect detailed information but only over areas of interest that were detected and logged during the first runway scan. This step is complete after the UAV has conducted its second scan and the data has been processed.

After the first and second scan have been completed and all data has been processed and formatted, estimation of required repair materials can be completed and repair operations can commence.

C. GENERAL SYSTEM SET UP

For this project, a DJI Phantom 4 Pro was selected for its high-resolution camera, max flight range, and flight time in conjunction with a 128GB SD card for data storage. Further characteristics of the Phantom 4 are discussed earlier in Chapter II. For a tablet interface, an iPad Pro10.5 with Wi-Fi and cellular capabilities was chosen for several reasons, important among them are:

- Access to various flying applications via the Apple app store
- Its large, bright and high-resolution Retina Display®
- Its ready fitness for the Phantom 4 Flight Controller
- Its GPS capability
- A10X chipset

The iPad Pro was used in conjunction with a standard Phantom 4 Pro flight controller. In addition to the iPad Pro and flight controller combination, a DJI Flight Controller with an integrated screen, that comes default with Phantom 4 Pro+ models, was also used. The processing center, where the neural network was trained and where all post-UAV flight image processing was conducted, consists of a desktop computer with Windows 10 Pro 64-bit, Intel i7-7820X @ 3.60 GHz CPU, 8 cores/16 thread, 64GB RAM, 512GB SSD, and an NVIDIA GeForce GTX 1080 Ti GPU.

D. INITIAL RUNWAY SCAN

The initial scan of the runway is comprised of five main elements: flight over the runway, data transfer, automated damage detection, damage localization, and mosaic creation. These elements occur sequentially and are visualized in Figure 14. The overall goal of the initial runway scan is to quickly provide a whole look of the runway in order for the EOC to quickly determine MOS.

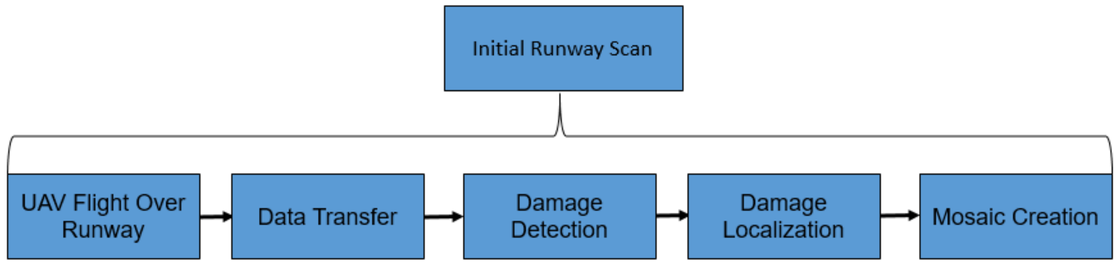


Figure 14. Design overview of initial runway scan architecture

1. UAV Flight over the Runway

The first step in determining MOS is to collect images of the runway with high enough resolution to accurately detect and classify various pieces of damage but with as few images as possible to decrease processing time. In order to create a mosaic of the entire runway, a series of overlapping images is needed so that they can be stitched together. Two flight methods were tested in order to collect the required data: automatic flight and image capture based on pre-planned waypoints and manual flight with live video feed.

In order to conduct automatic survey waypoint mapping, a software called Pix4Dcapture (Pix4Dcapture, n.d.) was used. Pix4Dcapture is a mobile application that is used to plan flight routes designed to collect data for 2D and 3D mapping (Pix4Dcapture, n.d.). This software offers a straight-forward graphical user interface (GUI) that allows users to simply draw a box over an area that they want surveyed and adjust various flight parameters such as altitude, speed and image percentage overlap. Of note, we tested ‘fast’ and ‘slow’ speeds which determine how fast the UAV will be flying when it takes an image.

With a designated area and flight parameters set, Pix4D automatically plans the optimal flight path and interfaces directly with the UAV for automatic flying and image capture.

Pix4D also offers the ability to preplan and save missions with specific parameters for future execution and does not require an active Internet connection to be used. While the drone is in flight, if the battery level gets to be too low, the drone automatically returns to the position from which it took off. If a mission space is too large to be completed on one battery, it can be paused and resumed after a new battery has been installed. A look at the Pix4Dcapture GUI can be seen in Figure 15. Pix4D allows a quick and effective way for a user to simply draw a box over an airfield, specify required parameters and let the system handle the rest.

In order to do manual joystick-based controller flying, DJI GO 4 was used. DJI GO 4 is the official app produced by DJI in order to pilot the Phantom 4 Pro. Although there is no direct way to do preplanned waypoint mapping, the app offers the ability to manually take pictures and/or video while in flight.

Regardless of app used to fly the UAV, the flight paths used were set up so that the UAV started at one end of the runway, was able to capture the entire width of the runway, and proceeded down the runway until it reached the opposite end.

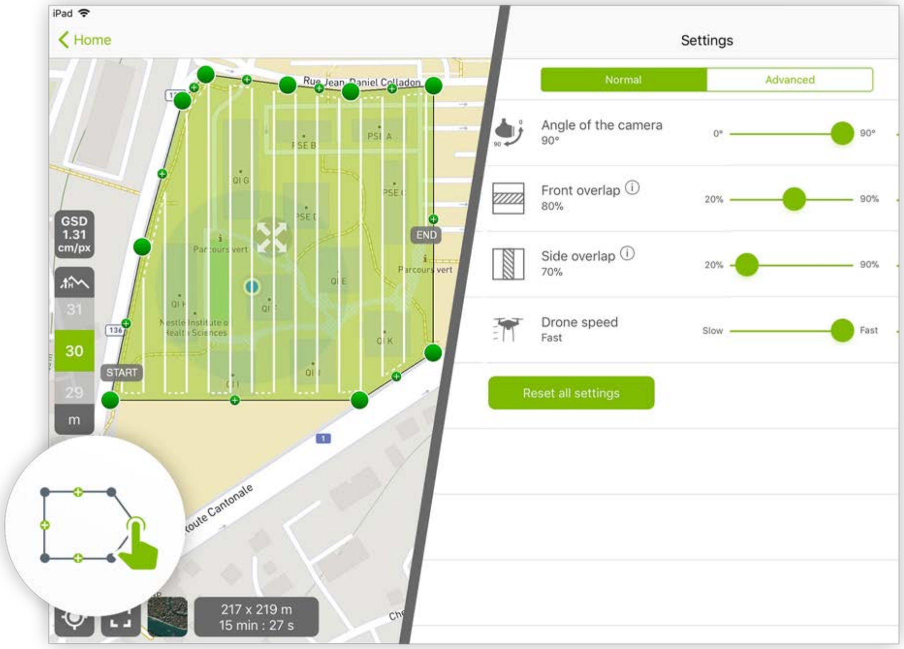


Figure 15. Pix4D user interface. Source: Pix4D (2018).

2. Data Transfer

Two variations were tested in order to collect pertinent data: post-flight image transfer and live video stream via the DJI GO 4 app.

Whether using automated image capture via Pix4Dcapture or manually collecting images with DJI GO 4 app, the captured images are stored locally on the drone on an SD card. In order to transfer the high-resolution images off the drone, users must wait until the drone is no longer actively being flown and is either hovering or landed. When using a tablet and standard flight controller, this data transfer can be accomplished wirelessly via Wi-Fi. However, this transfers the files to the tablet interface in use with the flight controller. The images would then have to be transferred to a local processing center for analysis and tagging. Transferring a large number of high resolution (7MB) images via Wi-Fi twice can be very time consuming and is subject to communication slowdown due to environmental degradation and interference. Another option, regardless of flight controller set up is to manually remove the SD card from the drone and directly upload to a processing center computer. With the use of a USB 3.0 SD card reader, this is generally a much faster approach. Of note, when using a tablet and standard flight controller configuration and capturing images, the system can be configured to store a low resolution (500KB) version of the images into the tablet cache. However, in addition to being lower resolution, the images do not contain any metadata information.

In addition to post-flight image transfer, live video stream was also tested. While manually flying, DJI GO 4 app has a constant live stream video feed that broadcasts from the drone over standard 5.8GHz or 2.4GHz Wi-Fi. By adding an HDMI output module onto the standard Phantom 4 flight controller, this live video can be output to an external monitor via HDMI cable. The HDMI output capability come standard with the ‘plus’ model flight controller. An HDMI video capture to USB 3.0 adapter was then used in order to feed the live stream into a laptop or local processing station. The overall system set up is depicted in Figure 16. In order to prevent overheating during prolonged use of the USB 3.0 capture adapter, a series of Aluminum Heatsink Cooler Circuit Board Cooling Fins were placed on both sides of the dongle. This setup allows for live video capture on either the standard flight controller or the “plus” model flight controller. With a live video feed, high definition images (1920 x 1080 pixels) can be captured and saved directly at the local processing center in near real time by capturing frames of the video feed. The capture of the frame rate can be tuned to match drone flying speed in order to get the correct amount of image overlap.

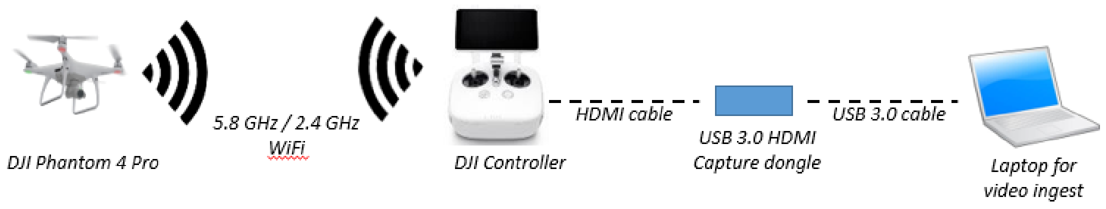


Figure 16. Live video data feed system setup.

3. Damage Detection

Whether it is a frame from a live video or an image taken and later transferred via SD card, each image is analyzed for airfield damage by using a trained neural network. The goal of this step is to detect if there is some damage in the picture, determine what type of damage it is, and where in the picture it is located. It is not enough to simply state that there is damage in the image; the location and number of instances of damage are also very important. The need for localizing and classifying numerous areas of damage in a single image means that image classification will not work and object detection is needed. Before an object detection model can be trained and used, a large data set of images that are representative of the actual data expected must first be compiled. With access to a large data set, those images must be manually labeled with location and classification of the damage type. Only then can a model be trained on the data set.

a. Gather and Format Data

Data was gathered from two different sources. The first source is ImageNet. ImageNet is a database that houses a multitude of images sorted by category (ImageNet, 2016). If there is a lack of direct access to pictures of damaged airfields, ImageNet's pictures of various sized potholes can be used for training. Although not quite the same in nature, a pothole on a road is somewhat similar to a crater on an airfield. However, there are some major differences between the two including area debris, which is usually present in airfield damage but not with potholes. These differences can impact the capability of a trained object detection model. Two examples of images of potholes from ImageNet can be seen in Figure 17. The images used from ImageNet vary in resolution, image size, angle taken, and what percentage the pothole takes up in each image. In addition, they are not completely representative of damaged airfields since many of them include various objects in the background and foreground such as vehicles, sidewalks, people, etc. These background objects are not expected to be present in images taken of an actual damaged airfield.

In addition to ImageNet, the Air Force Civil Engineering Center (AFCEC), located out of Tyndall Air Force Base, Florida, provided a massive amount of data from their testing and training on ADR operations. Due to the high resolution of the data provided by AFCEC, the images were broken down into numerous smaller images using a sliding window technique. A window of size 1000 x 600, starting in the upper left-hand corner, was slid across the images with a 50% overlap in the horizontal and vertical directions. A visualization of this technique can be seen in Figure 18. Smaller images were used in order to reduce time and memory requirements for training the model. This contrasts with the images used from ImageNet that were used directly without any modification or pre-processing.

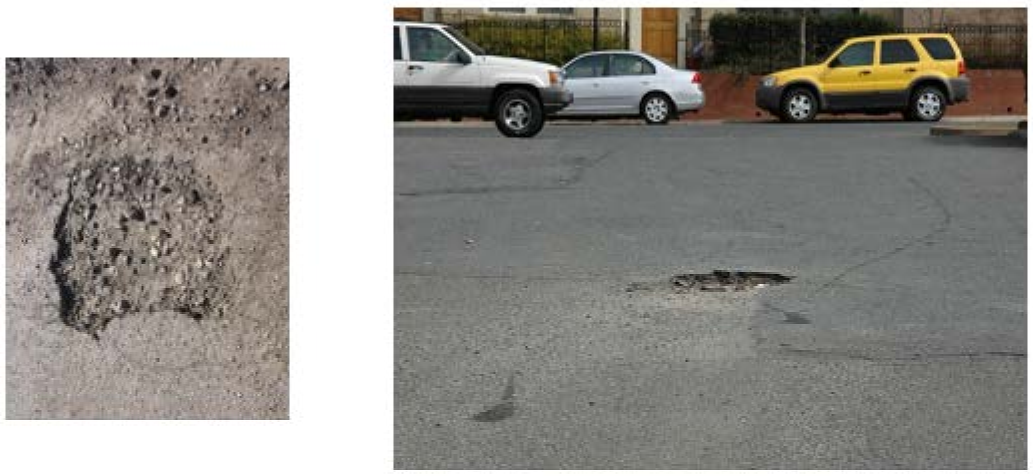


Figure 17. Examples of varying pothole images from ImageNet
Source: Pothole Synset (2010).

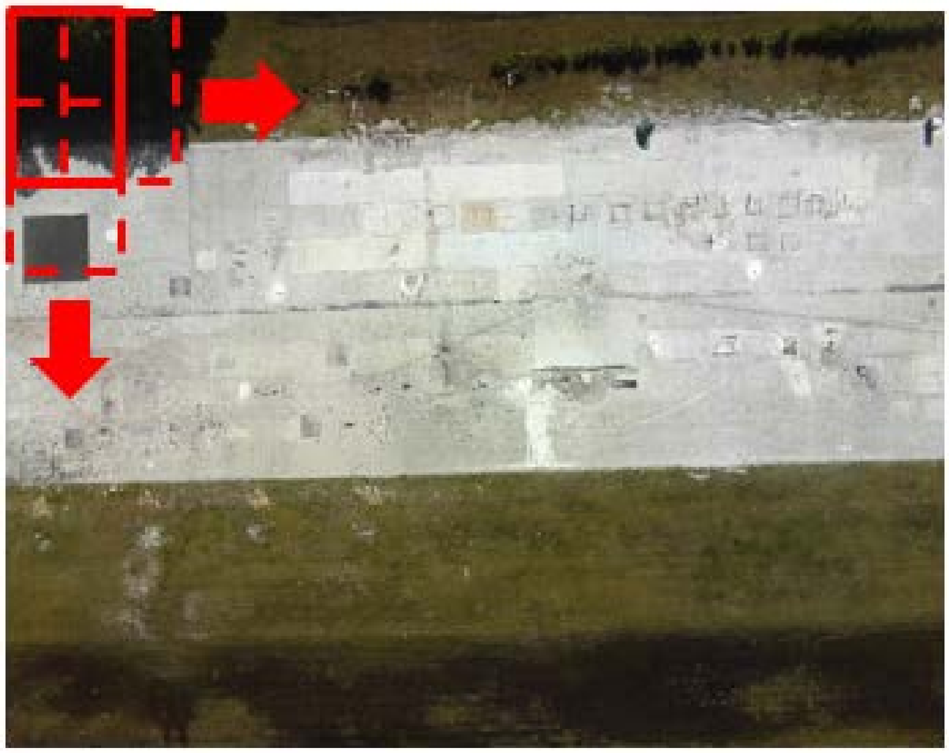


Figure 18. Sliding window technique.

b. Label the Data

With data gathered, each image had to be labeled using bounding boxes to identify where the damage was located in the picture and classified appropriately. This was accomplished using a program called LabelImg. The interface for LabelImg can be seen in Figure 19. Each image is loaded into the program individually, a user manually drags a bounding box over the area of interest, the bounding box is assigned a class or group, and the image is saved. Saving creates an XML file containing the data of the original image, the bounding box locations within the image, and the class or group assigned to that bounding box. These XML files are in PASCAL VOC data format. However, TensorFlow requires TFRecords in order to train models. Consequently, we generate TFRecords from the previously produced XML files. This format transformation was done using code adapted from Tran’s tutorial on developing a raccoon detector using TensorFlow (Tran, 2017).

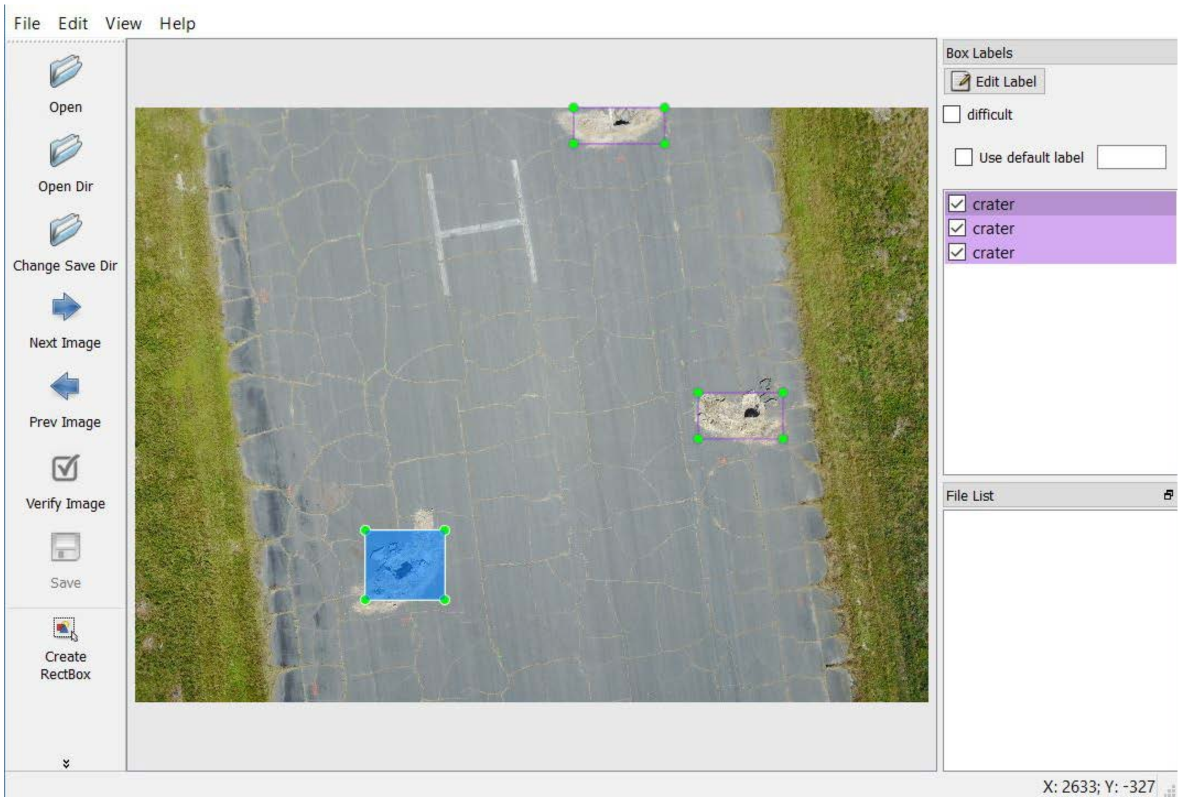


Figure 19. Interface for LabelImg program.

c. *Training the Model*

In order to train the neural network model, a virtual environment was made using Anaconda. The packages used in the virtual environment are listed in the Appendix.

TensorFlow offers both a CPU- and GPU-based installation guide. Although the CPU version is easier to install, the GPU version offers greater speed in training and use. A CPU and a GPU version each were tested, and findings are discussed in detail below. In order to use the GPU version, the following were also installed onto the system: NVIDIA 398.36 video driver, CUDA toolkit 9.0, CUPTI, and cuDNN.

Two different models were trained and tested using TensorFlow. The first model used TensorFlow’s SSD Inception V2 configured for the Pets dataset. This model was configured to be a single class damage detector and was trained on 388 instances of “damage” consisting solely of un-modified images of potholes acquired from ImageNet.

The set was randomly split into 80% for training and 20% for validation. This model was trained for a total of 698 training steps.

The second model was built to be a dual class detector for both "spalls" and "craters." To meet the requirement that we search numerous pictures quickly with a high degree of accuracy, we used TensorFlow Faster Region-Based Convolutional Neural Network was used in conjunction with the Resnet101 feature extractor for training. With 913 instances of craters and 944 instances of spalls, all of which were acquired from AFCEC during actual airfield damage testing, each set was randomly split into 80% training and 20% validation groups. The training images were then grouped together, as were the validation sets. This split was applied to each class individually in order to ensure that the final random sampling would include an equal distribution of spalls and craters in an effort to avoid a bias toward one class over the other. The model was trained for 223,469 training steps.

The randomness of the sampling in order to achieve a 80%/20% split between each class was conducted by first converting all of the information from the XML files into a CSV format using code adapted from Tran's tutorial on developing a raccoon detector using TensorFlow (Tran, 2017). With all the data in CSV format, it was then randomized using Microsoft Excel's built-in randomize capabilities.

d. Using the Model

We applied our trained models to detect and classify damage in newly collected images. The model is employed by establishing a TensorFlow session, loading each image in turn and analyzing it individually. When an image is analyzed, the system returns locations of potential areas of interest, the classification of that area of interest, and an overall confidence score for that area. Only areas with confidence of greater than or equal to 50% were considered. This information is saved for later use and computation. In addition to numerical outputs, the original image is returned with colored bounding boxes over the areas of interest. Classification and percentage confidence is also displayed. This output image, with bounding boxes over detected damage, can then be saved separately from the original image for later processing. Figure 20 is an example image that is fed into the system and Figure 21 is an example of that same image after the system has conducted its analysis.

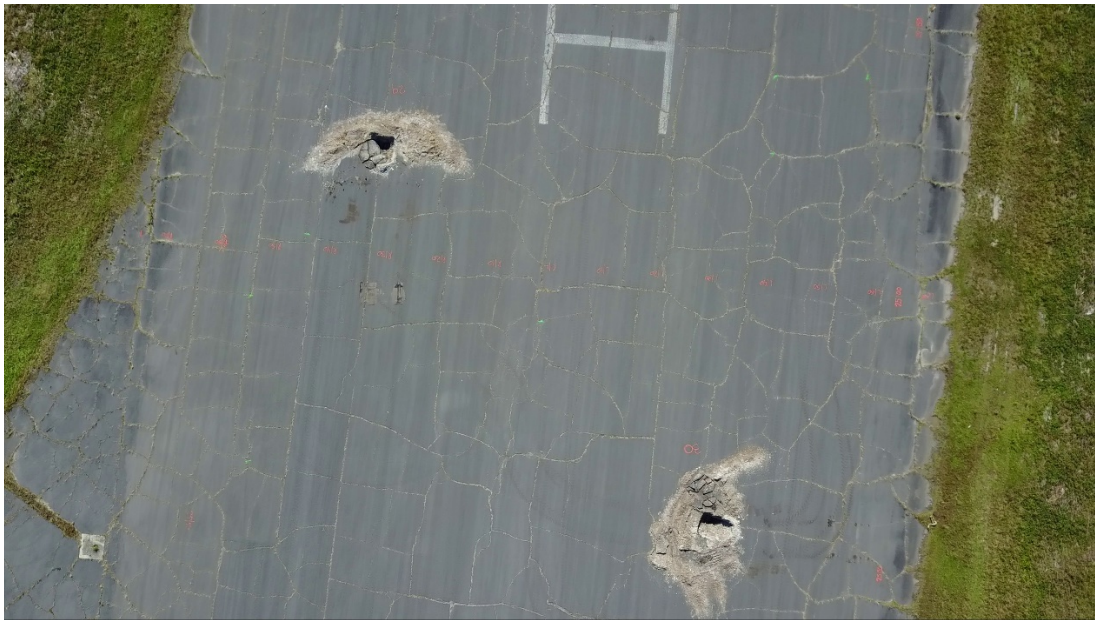


Figure 20. Original image input into the system.

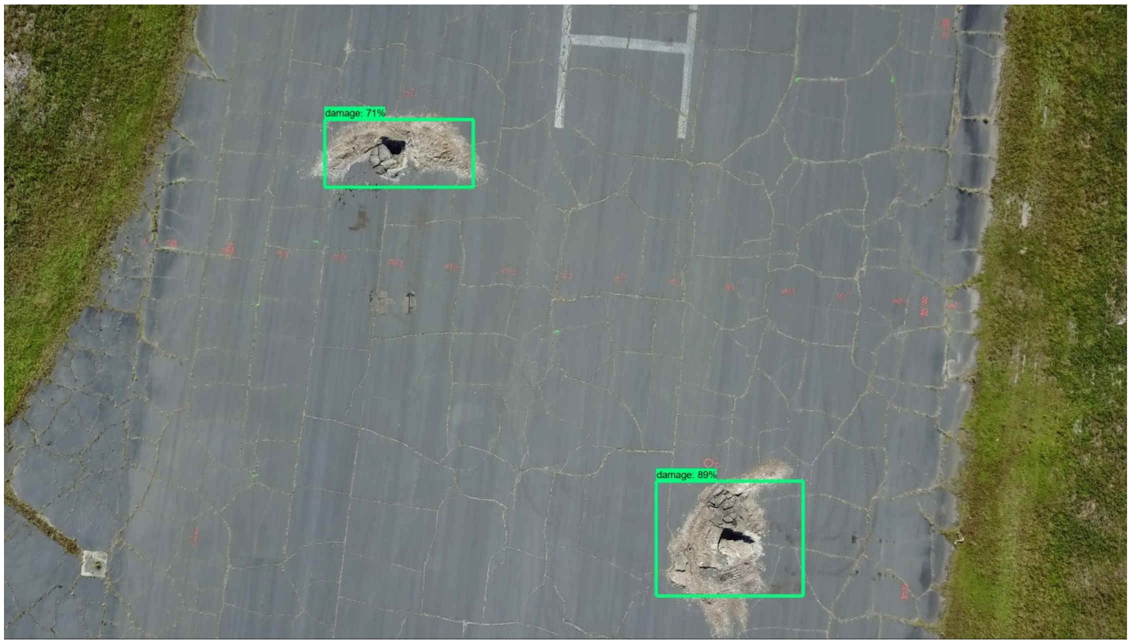


Figure 21. Example output of analyzed image.

4. Damage Localization

With each of the images analyzed and all damage detected and labeled, the next step is to extrapolate real world coordinates of the location of the damage detected in the image. This is a multi-step process, which is shown in Figure 22.

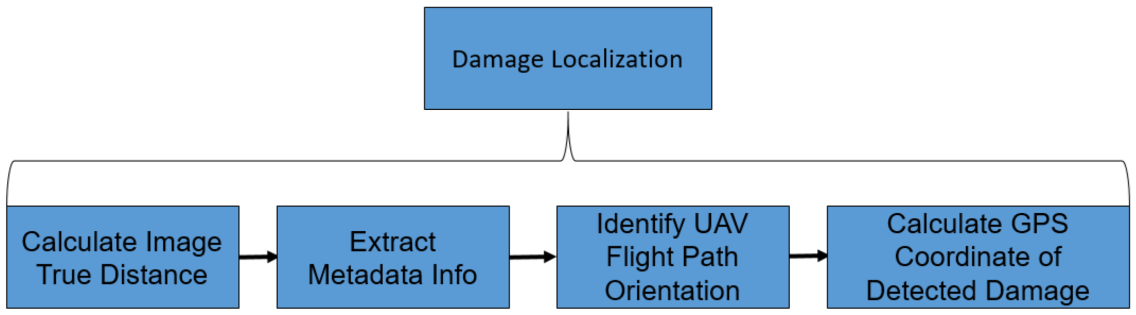

Damage Localization
Calculate Image True Distance
Extract Metadata Info
Identify UAV Flight Path Orientation
Calculate GPS Coordinate of Detected Damage

Figure 22. Overview of damage localization system design.

Based on UAV height and camera parameters, it is possible to determine true ground distance per pixel in an image. This is known as Ground Sample Distance (GSD). With a known GSD, true distance can be calculated based on number of pixels in height and width of an image. An overview of how this is calculated can be seen in Figure 23 and the actual GSD calculation used are shown in Figure 24. Note that this is camera/sensor dependent and will have to be adjusted for different equipment and UAV altitude when collecting data. However, with a constant set up, the calculation is only required to be done once as the equipment and image size will remain constant. If it is necessary to recalculate GSD, it is a very fast calculation and can easily be repeated.

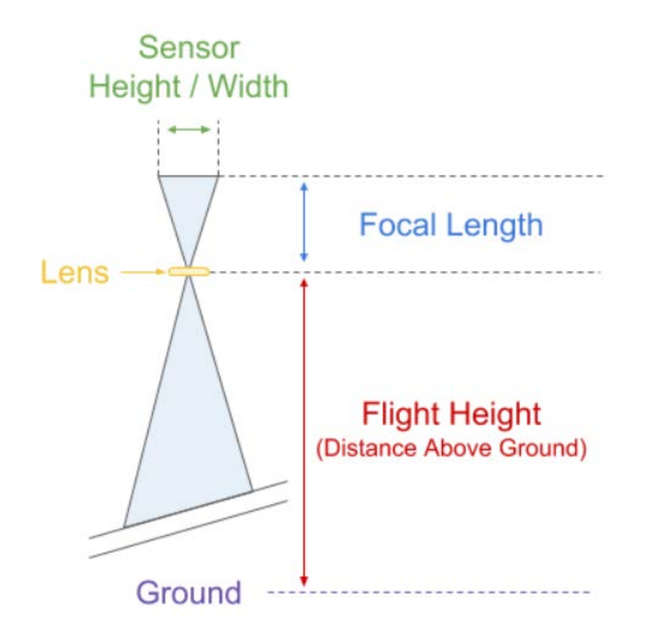


Figure 23. Visualization of GSD parameters. Source: What is ground sample distance (GSD) and how does it affect your drone data? (2018).

$$GSD_h = \frac{Flight\,Height * Sensor\,Height}{Focal\,Length * Image\,Height}$$

$$GSD_w = \frac{Flight\,Height * Sensor\,Width}{Focal\,Length * Image\,Width}$$

Figure 24. Equations to calculate GSD. Source: What is ground sample distance (GSD) and how does it affect your drone data? (2018).

Every image taken by a UAV has various pieces of information stored with the image in the metadata. This includes details such as camera make, camera model, F-stop, etc. In addition to camera parameters, UAVs log their GPS coordinates in every image taken. When looking directly down, these GPS coordinates correspond to the center point of the image. To extract this metadata, we used a publicly available open source Python script (maxbellec, 2018).

By examining the first and last image in a data set, which correspond to the first and last images taken and also the ends of the runway, we can determine the overall orientation and flight direction of the UAV. This information is critical for determining true location of damaged areas detected in an image instead of just relative location. For example, if damage is detected along the left border of an image, based on the overall orientation, this might actually be the most eastern part of the image if the UAV had been capturing images while flying due south.

Using the known values of true ground distance covered in each image, the GPS location given in each image, overall orientation/direction of the data set and the relative location of the damage in the image, a true location of the damage can be determined in the form of a GPS coordinate. This process is detailed in Algorithm 1.

```
FIND GPS LOCATION OF DAMAGE
extract meta data from first image in set
extract GPS coordinate from metadata
extract meta data from last image in set
extract GPS coordinate from metadata
calculate angle between the first and last GPS points

for each image in data set
    extract metadata of image
    extract GPS coordinate from metadata
    for each damage detected in image
        calculate angle to damage in image
        calculate distance from image center point to damage
        correct for overall orientation angle
        calculate GPS coordinate of damage using GPS coordinate of image, distance,
             and angle to damage
        table look up to determine what type of damage was detected
        save GPS coordinate and type of damage
```

Algorithm 1. Pseudocode to determining GPS location of detected damage.

5. Mosaic Creation

Once all the images are analyzed and marked, they are rotated and stitched together in order to create a mosaic of the entire runway. Since we know that the UAV always starts at one end of the runway and proceeds to the other, the natural orientation of the images is a bottom-to-top scheme. If they were to be stitched together in this fashion, a very tall image would be produced. By rotating the images and then stitching them together, we produce a left-to-right orientation scheme, much like a panorama image. In order to rotate the image without any sort of cropping or distortion, we used a method developed by Remi Cuingnet (Cuingnet, 2016).

The images are stitched together using the Python OpenCV library. Once the mosaic is created, grid lines are overlaid onto the image every 10 meters in order to give a better sense of distance covered in the mosaic. In addition, these grid lines can assist in the determination of MOS. These grid lines are applied to the mosaic by using previously calculated overall true distance in the mosaic based on the GPS coordinate of the first and last image and using the Python OpenCV library to draw evenly spaced lines. For example, if the total distance in the mosaic was 200 meters, in order to achieve a line every 10 meters, a line would be drawn every 5% of the length of the mosaic. Figure 25 shows an example mosaic produced from a series of single images over a particular runway taken from a UAV before any damage detection. Figure 26 shows the same images including damage detection with damaged areas now highlighted via a green bounding box. Note that because the UAV was slightly offset when taking the images, the final mosaic has a downward slope effect. The implications of this slope effect and the accuracy of grid line placement is discussed in Chapter IV. The slope effect can be avoided for a cleaner looking mosaic and accurate grid line placement by ensuring that a UAV flight path is as straight as possible during the waypoint planning and data collection phases or by conducting additional post processing. Figure 27 is an example of a mosaic with grid lines overlaid and labeled every 10 meters.

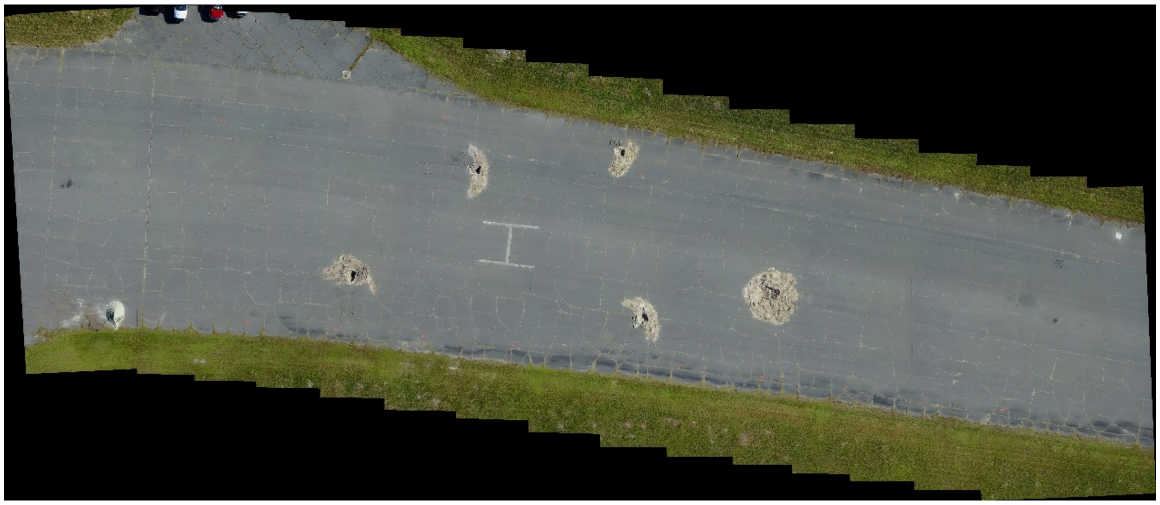


Figure 25. Example mosaic of a runway produced from a series of images taken from a UAV.

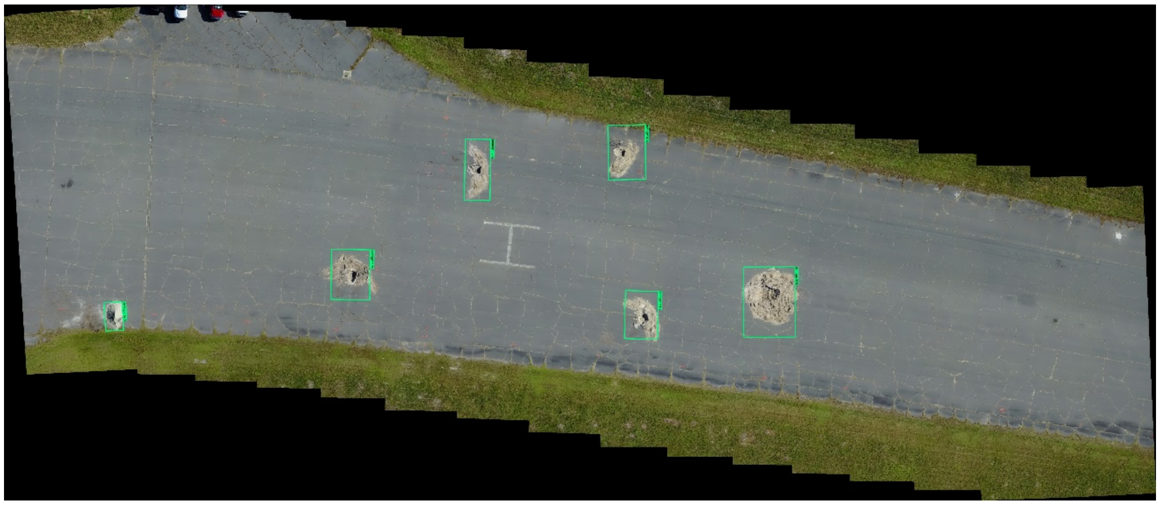


Figure 26. Example mosaic of a runway produced from a series of images taken from a UAV with automatic damage detection.

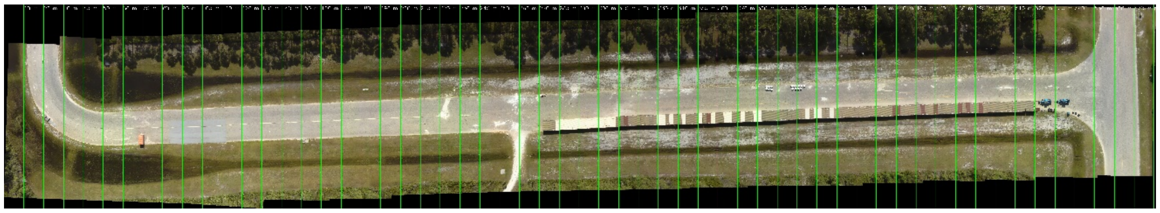


Figure 27. Example mosaic of a runway on ramp produced from a series of images taken from a UAV with grid lines overlaid every 10 meters based on GPS data.

The completion of the damage detection and mosaic concludes the initial runway scan. At this point, personnel at the EOC should have enough information in order to determine the MOS or if the runway is even usable. Once it has been determined that the runway is worth fixing and an MOS is selected, the UAV can be sent out in order to conduct a more detailed look only in areas of interest such as damage within the selected MOS.

E. SECOND RUNWAY SCAN

The goal of the second runway scan is to gather more information but only in specific areas that were detected and logged after the first runway scan. This is done separately because in order to get more detailed data, including the necessary requirements to build 3D models, more data (i.e., images) must be collected and more processing power is required in order to produce the models. This is not feasible, nor is it needed, in a real-time manner for the entire runway. Using GPS locations of damaged areas saved from the initial runway scan and a specific MOS selected at the EOC, the UAV can be directed to only collect the data required for 3D models in areas of interest instead of the entire runway saving valuable time and processing speed.

To produce 3D models and volume measurements, we must first collect additional imagery. In order to create 3D models from 2D imagery, multiple images are required over an area of interest with an overlap in imagery in both the x and z axes. For this purpose, we again used Pix4Dcapture as it offers the ability to switch from a single grid, which is good for image stitching and mosaic creation, to a double grid, which is better for 3D model creation. A visualization of the differences between 2D and 3D grid options can be seen in Figure 28.

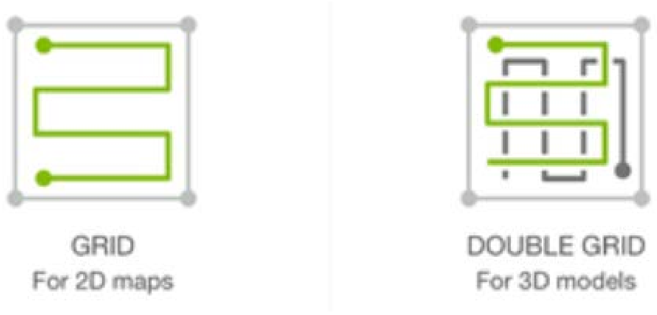


Figure 28. 2D and 3D grid options in Pix4D for mission planning

Once the images have been captured and transferred to the local processing station, either wirelessly or via SD card swap, as discussed earlier, they can be used to create a 3D model. We evaluated two different methods of creating the 3D models. The first was OpenDroneMap. OpenDroneMap is an open source toolkit that has the tools necessary to transform 2D images from UAVs into 3D models (OpenDroneMap, 2018). These 3D models include point clouds, digital surface models, and digital elevation models. Although OpenDroneMap is a command line-based program, there is a free GUI extension called WebODM that runs in a browser window and offers a simple interface that allows the user to take advantage of OpenDroneMap capabilities. We installed WebODM using a Docker container in order to package the code and all of its dependencies (What is a container, 2018). Once Docker and WebODM are installed, a new project can be created by selecting "Add Project," adding all of the images taken during a particular mission, setting the various available options and selecting "Start." After the processing is complete, all of the assets and models can be downloaded or some may be viewed in the browser itself such as a 2D/3D model viewer of the area and also a 2D rendering of the area overlaid onto Google earth. The 3D model viewer has multiple built in tools such as a textured profile option and height profiler.

In addition to OpenDroneMap/WebODM, we also tested Agisoft PhotoScan was also used. Similar to OpenDroneMap, Agisoft PhotoScan is a product to convert 2D images taken from UAVs into various 3D models including dense point clouds and digital elevation models which can then be used to calculate volume and distances between points (PhotoScan, 2018). Agisoft PhotoScan also offers the capability to balance accuracy and speed during model creation. In addition to manually choosing which models to construct, pre-defined workflows can be established in order to automate and easily repeat the production process limiting the required amount of user input. Once the models are created, they can be saved individually for further editing and manipulation. An example output of a 3D model produced using Agisoft PhotoScan can be seen in Figure 29.

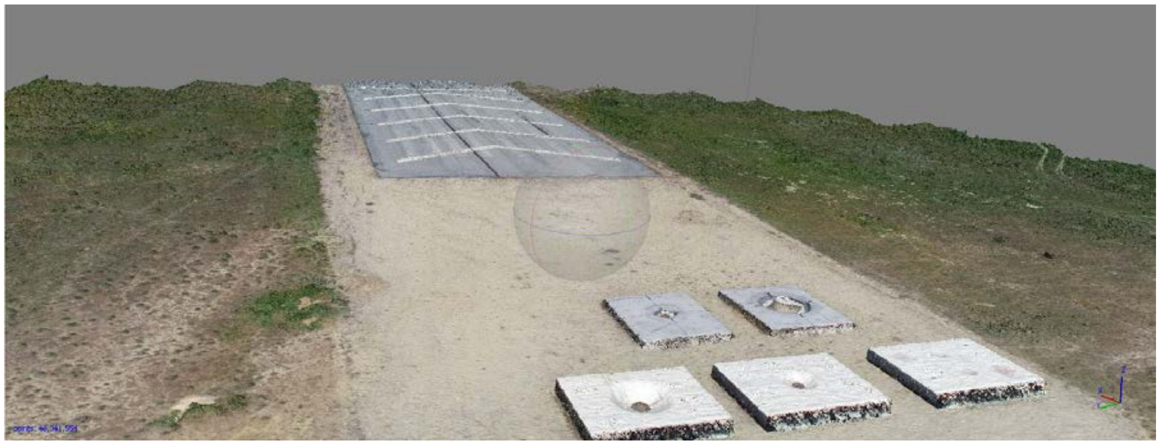


Figure 29. Example output of a 3D model produced with Agisoft PhotoScan.

F. REPAIR PLANNING

After the models have been created, they can be ingested into planning tools such as GeoExPT, the custom plug in for AutoCAD specifically designed for airfield operations. GeoExPT/AutoCAD has the ability to ingest many of the previously created models which can then be used to calculate size and volume of spalls and craters in order to make estimates on required repair materials.

This process concludes once the required models have been loaded into GeoExPT/AutoCAD and repair estimates have been made, at which point repair operations begin.

G. CHAPTER SUMMARY

This chapter has provided a detailed look at the methodology and system architecture used to develop our ADA system. It began by describing the design of the overall system including the initial and second runway scan. It then described each of the numerous sub-components that make up the system including survey mapping, mosaic creation, neural network training and use, and 3D model creation. In addition, this chapter listed the hardware and software used for system development and testing. Chapter IV discusses both findings and results from implementation and testing of the system.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPLEMENTATION AND TESTING

This chapter discusses the findings and results during implementation and testing of the system. While the overall system architecture did not change, numerous subcomponent changes were made for improved system performance.

A. UAV DATA FLOW

Numerous observations regarding how the data was transferred from the UAV to the processing station were made during the testing of this system. Some of major findings that directly impacted data flow are detailed below.

1. Live Video Stream

Initial testing was performed using the live video stream capabilities of both the tablet/standard flight controller configuration and the Pro+ flight controller while using the DJI GO 4 application. As discussed in Chapter III, with a live video feed being captured by the processing center, frames can be captured and saved using Python OpenCV packages effectively emulating the data flow process of taking a series of images while saving the time requirement of transferring them via Wi-Fi or SD card. However, there is no direct access to GPS location during this process. While video is being recorded, GPS coordinates are embedded into the video but this information is only being stored locally on the drone's SD card. In addition, there was no native way to directly get the GPS location of the drone while it was in fight. Breaking live video into frames would allow for mosaic creation and automatic damage detection processes while saving the data transfer time. However, there would be no grid lines or damage localization which is critical information for determining MOS. Consequently, live video capability was not further developed.

2. Flight Controller Preference

While flying the UAV utilizing the Pro+ controller with integrated screen, there is no way to install third party flying applications onto the device. Users can only use the DJI GO 4 application in order to fly the drone. In addition, as discussed in Chapter III, the DJI

GO 4 application does not have a direct way to do preplanned waypoint flying. In contrast, while using the iPad with regular flight controller, DJI GO 4 can be used for manual flying in addition to third party applications such as Pix4Dcapture for preplanned waypoint flying. Consequently, although the Pro+ flight controller was tested successfully in flying the UAV manually via the DJI GO 4 application, it was largely abandoned for the rest of the testing of the system in favor of the more capable iPad and standard flight controller set up.

3. Caching of Image and Video Data

As discussed in Chapter III, while the UAV is taking pictures that are being stored on the local onboard SD card, the system can be set up so that lower resolution versions of the same images can be stored on the iPad. This is true for video as well. However, these lower resolution images and video do not contain any metadata. This lack of metadata, specifically lack of GPS data, is a critical loss for post processing when trying to determine the location of damage. Using lower resolution images without metadata, an image mosaic can be created and the automatic damage detection could still be run but there would not be any localization of the damage or grid line placement which is a requirement for determining MOS. So, although the data can be gathered almost instantly on the local tablet controller while the UAV is in flight, we determined that the high-resolution images with the attached metadata would be required. This requirement necessitates a post flight image transfer after every flight. This transfer could be conducted via Wi-Fi or SD card swap while the drone was landed or in a hovering state using either DJI GO4 or Pix4Dcapture applications. This approach has an added benefit—since the data is not being wirelessly streamed, it cannot be intercepted by an adversary.

4. Wi-Fi and SD Card Data Transfer

Concerning the post flight image transfer, both Wi-Fi and SD card transfer were tested successfully. Using Wi-Fi, once a flight is complete, all images can be transferred to the iPad automatically. With the iPad and local processing center on the same network, those images could then be transferred over to the processing center for further production. Although transferring the media over Wi-Fi was successful, it was very slow relative to the transfer using the onboard SD card. When the drone has reached the end of the runway and all necessary data collection is complete, it can fly max speed back to the starting point where presumably the processing center would also be located. The onboard SD card then must be manually removed and transferred onto the processing center via a USB 3.0 SD card reader. This method proved to be much faster than the two-time transfer of the same data via Wi-Fi.

B. WEBODM AND AGISOFT PHOTOSCAN

Both WebODM and Agisoft PhotoScan were used in order to produce various 2D and 3D models from the images taken from the UAV. At the time of system testing WebODM did not support GPU acceleration whereas Agisoft PhotoScan does. In addition, GPU acceleration can be parallelized for even faster model generation. Puget Systems, a custom computer vendor, ran multiple tests and concluded that there was a significant decrease in computation time when using two GPUs and a smaller but noticeable decrease when using three GPUs (George, 2018). Being open source with an active community, OpenDroneMap could very well support GPU acceleration sometime in the future. However, because model generation time is a high priority, Agisoft PhotoScan was favored during system testing

C. CPU AND GPU TENSORFLOW

During initial system testing, the CPU only version of TensorFlow was used for model training. With an Intel® Core™ i7-7700HQ 500 training steps took roughly 11 hours to complete. However, the system was later updated to use the GPU enabled version of TensorFlow. With a NVIDIA GeForce GTX 1080 Ti GPU 223,469 training steps were completed in approximately 32 hours at a rate of roughly 1 step every 2 seconds. In addition, the GPU version of TensorFlow also provides the ability to use the GPU when using the trained model in order to conduct the actual detection. Although there is not much of a noticeable difference when using the model with a GPU instead of a CPU on a single image, there is much speed to gained when conducting damage detection over lots of images. Using the GPU, the model was able to conduct automated damage detection on 5,000 images sized 1000 x 600 in less than 20 minutes. Although there are extra requirements in order to install and configure the GPU version of TensorFlow, the benefits in model training and use far outweigh the cost.

D. POTHOLE DETECTION VS TWO-CLASS DAMAGE DETECTION

The initial single class TensorFlow model that was trained using unmodified images of potholes obtained from ImageNet was tested against simulated airfield damage consisting of damage caused by heavy machinery, and also against realistic airfield damage caused by explosives. Interestingly, the pothole-based single class damage detector model performed well on images of simulated airfield damage, but it did not perform well on realistic airfield damage. An example of the former can be seen in Figure 30 and the latter in Figure 31. In many cases, the single class damage detector model did not detect any damage even when tested against images of airfields riddled with damage caused by explosives.

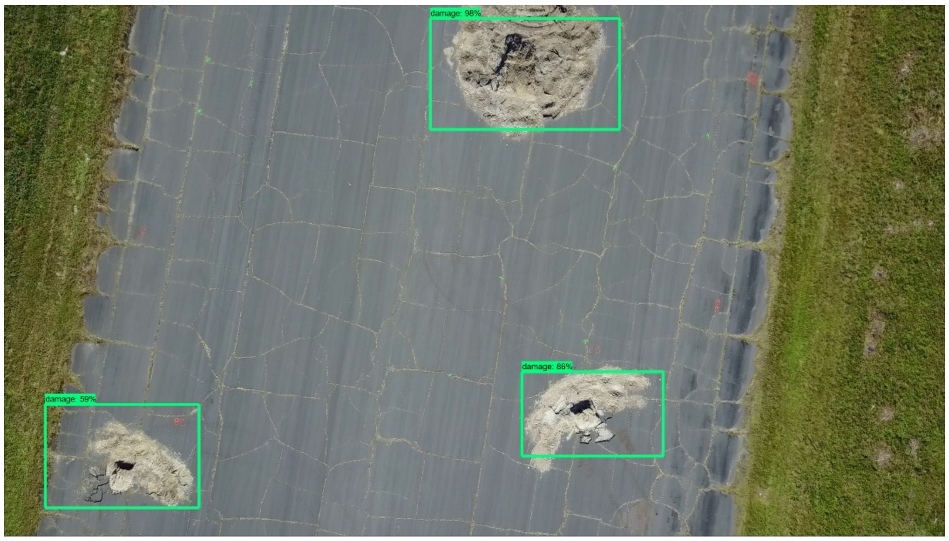


Figure 30. Single class damage detection on airfield damage caused by heavy machinery.

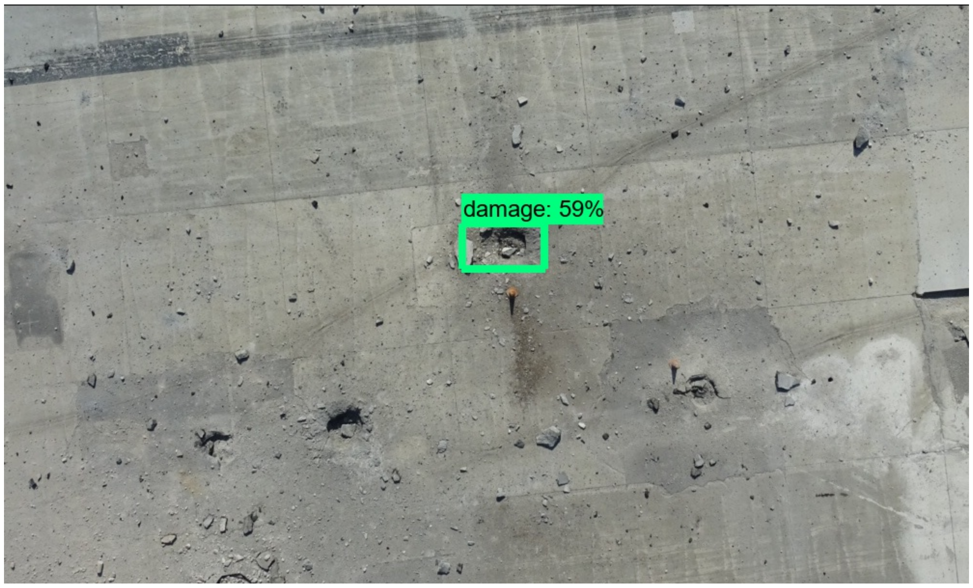


Figure 31. Single class damage detector model on airfield damage caused by explosives.

The poor detection rate is likely linked to the pothole images not being representative enough of actual airfield damage. In addition, in the simulated airfield damage, craters are surrounded by a light ring of dirt providing excellent contrast between the damage and the runway. Conversely, there is very little contrast in damage, debris, and the runway in images of airfield damage made by actual explosives. The high contrast in the simulated airfield damage would make it much easier for the model to identify the damage compared to the low contrast of the realistic airfield damage. Due to the poor performance of the single classification model on actual airfield damage, this model was discarded in favor of the two-class model (described in section 3.D.3.c) that was trained on actual airfield damage as the results were much better.

E. GPU TENSORFLOW AND IMAGE STITCH CONFLICT

During initial system testing using the CPU version of TensorFlow, there were no issues using the trained model and the Python OpenCV library to create the mosaic in the same Python file. However, using the GPU version of TensorFlow and the Python OpenCV library to create the mosaic was unsuccessful. This is likely due to a conflict in GPU memory allocation. This conflict occurred even when the TensorFlow session was closed out before the image stitching was initiated. Consequently, these features need to be run independently in separate files in order to avoid complication with the GPU.

F. UAV PICTURE TRIGGER MODE DURING IMAGE CAPTURE

One of the options using Pix4Dcapture while conducting image surveys is the ability to set the UAVs picture trigger mode to either fast or safe. In fast mode, the UAV will fly through the image waypoints taking an image as it goes along. In safe mode, the UAV stops at each waypoint in order to take the image. In addition, users can set how fast they want to the UAV to fly.   In an effort to decrease total survey time, the fastest settings were tested for both options. However, this leads to a “vanishing” effect in the mosaic. An example of this can be seen in Figure 32. This is likely caused by the fact that the faster a quadcopter UAV, including the Phantom 4 Pro used here, flies, the more of an angle it needs to tilt. When the camera is set to point at 90 degrees, which correlates to straight down while the UAV is hovering, this actually leads to an angled image when the UAV is in motion as the entire body of the UAV is now at an angle. Therefore, the slow trigger mode was used in conjunction with fast drone flying speed as this leads to consistent image angles resulting in better mosaics and a decrease in flight time as the drone will quickly move between waypoints. Alternatively, this problem could also be addressed by dynamically adjusting the camera angle, but the Phantom 4 Pro does not support this feature.

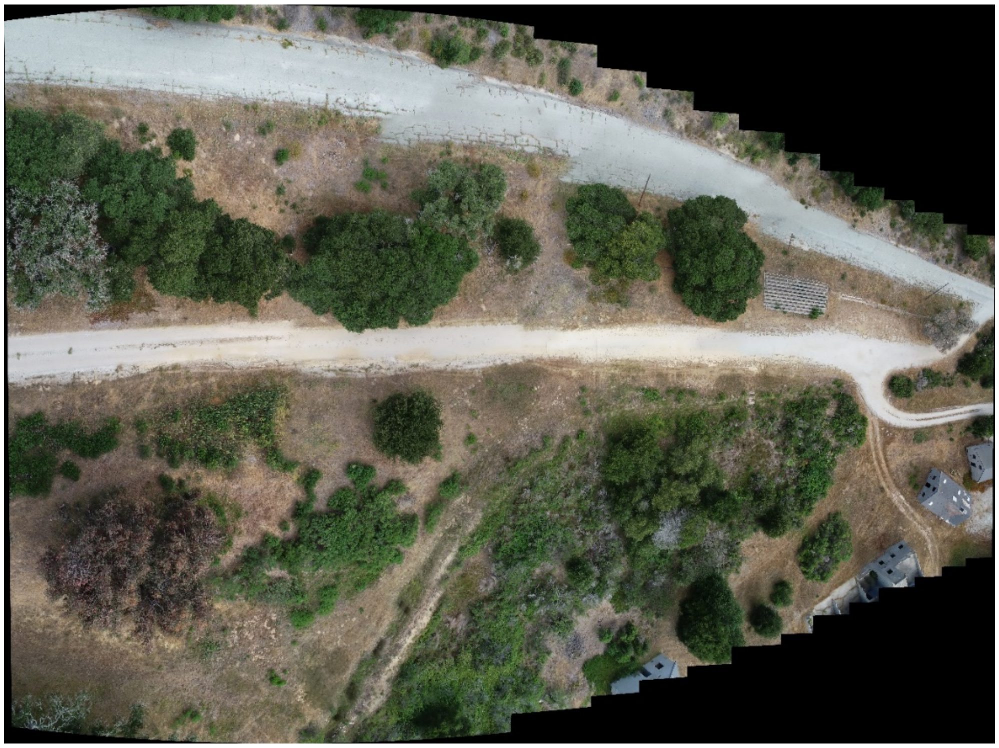


Figure 32. Mosaic showing vanishing effect due to UAV flight parameter settings.

G. RUNWAY IMAGE REQUIREMENTS AND IMAGE STITCHING SPEED

During the testing of the system, we measure the correlation between number of images collected and processing time required for mosaic creation. The results, shown in Table 1, were produced using the GPU enabled Python OpenCV library to conduct the image stitching and a NVIDIA GeForce GTX 1080 Ti GPU. Images were captured from a UAV and are highly overlapping, which allowed for testing of mosaic creation with different numbers of equally spaced images that cover the same ground distance. As the number of images decreases for the same ground distance covered, the percentage of image overlap is also decreased. Processing Time is strictly the time required to create the mosaic and does not include the tie to load images or save the created mosaic. Mosaic Quality refers to its accuracy and ability to be used in order to determine MOS.

Table 1. Relationship between ground distance, number of images, image stitch processing time and mosaic quality.

Ground Distance (m)	**Number of Images**	**Processing Time (sec)**	**Mosaic Quality**
270.61	7	28.67	Good
270.61	4	4.23	Good
270.61	3	2.36	Good
509.92	7	10.715	Good
509.92	5	4.57	Good
509.92	4	1.37	BAD

Table 1 is an example of how an effective mosaic can be created over a set distance using different numbers of equally spaced images. In addition, lowering the number of images for a set distance can yield positive results by lowering the required processing time without sacrificing usable quality. However, with too few images, as can be seen in the last entry of the data table, there is not enough image overlap for the image stitching algorithm resulting in an unusable product. In the case described above that resulted in a bad quality product, the mosaic was only a stitch of 2 of the 4 images. Visual representations of the mosaic created for the first and third entry can be seen in Figures 33 and 34, respectively, with grid lines placed every 10 meters. It is important to note that although they are very similar, the grid lines slightly differ from Figure 33 and 34. The reason for this variation is discussed in the following section.

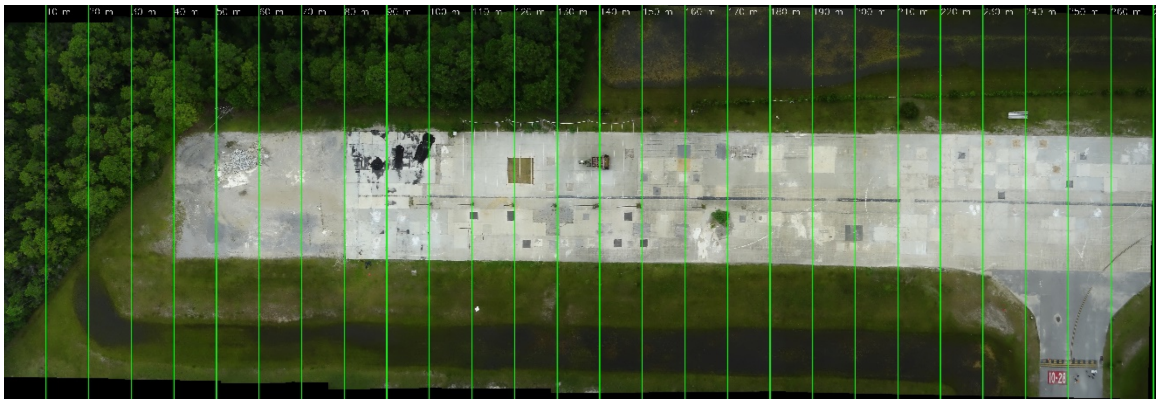


Figure 33. Mosaic created over a distance of 270.61 meters using 7 equally spaced images.

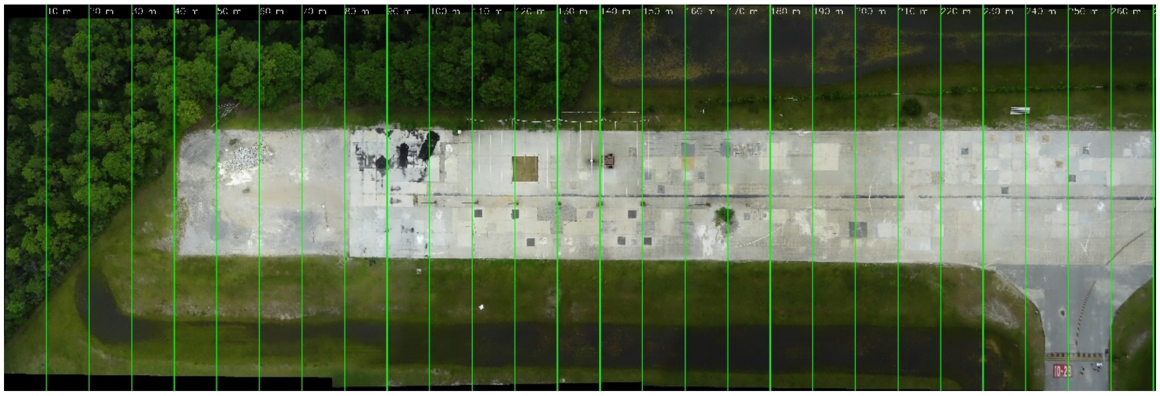


Figure 34. Mosaic created over a distance of 270.61 meters using 3 equally spaced images.

Table 2 shows the results from a similar experiment except with inclusion of different ground distances covered using the same number of images. Again, the images are spaced equally over the ground distance covered such that increase in distance results in a smaller degree of overlap.

Table 2. Relationship between varying ground distance, number of images, image stitch processing time, and mosaic quality.

Ground Distance (m)	**Number of Images**	**Processing Time (sec)**	**Mosaic Quality**
167.27	5	15.33	Good
350.64	5	6.24	Good
509.92	5	4.57	Good

The above information shows how using the same number of images over greater distance can decrease processing time while covering more ground and without sacrificing product quality.

When dealing with distances of multiple kilometers in distance, efficient management of image overlap is critical in order to minimize processing time while still producing a usable mosaic. Many factors will influence the image overlap percentage required for a set distance including lighting, how discernable features are for matching, camera resolution, and flight altitude.

H. GRID LINES ERROR DUE TO IMAGE ROTATION

It is important to note that current implementation of the grid laying system is based on an accurate and precise flight over the runway. As discussed in Chapter III, if the images are taken at a slight angle, a tilted mosaic is created. An example of tilted mosaic, due to UAV offset, can be seen in in Figure 33. Since the grid laying methods are based on the assumption that the images were taken straight on, there is no error handling to account for this offset. In addition, the amount of mosaic offset is directly influenced by the number of pictures used to create the mosaic. For example, even though Figure 33 and Figure 34 cover the same ground distance, the grid lines are slightly off from one another because they used a different number of images to create the mosaic thus resulting in one being slightly more tilted than the other. However, as long as the UAV maintains a constant course without any rotation or deviation to the sides and as long as the camera is fixed to be straight down when images are taken, an accurate mosaic and consequently, accurate grid lines can be applied to the image. It is possible to correct for this error with knowledge of UAV flight parameters such as angle and speed, but we leave that implementation for future work.

I. DETECTION OPTIMIZATION AND PERFORMANCE EVALUATION

The two-class neural network model, which detects both spalls and craters, was assessed on 311 instances of craters, 211 instances of spalls, and 1568 images of airfield images with no damage. These images were not included during the original training and validation phases of model training in order to avoid any form of predisposition of the model. In order to test the effectiveness of detection of the model, bounding boxes were manually drawn around areas of damage and classified using the previously discussed LabelImg program. This creates an XML file with the damage type and location which can then be checked against the location and classification results that the system produces. A detection was counted as a True Positive if both the classification label was correct and the location fell within an acceptable error threshold where error was measured as pixel distance between center of detected damage and center of ground truth location. We calculate the threshold based on percentage of pixel distance relative to the total resolution of the image. We tested acceptable percentage error of 5%, 10%, 15%, and 20%

During the testing of the two-class damage detector, multiple modifications were made to both the system and the data set in order to increase performance. In addition, the two-class trained model was tested as a single class damage detector in order to see if there was applicability in the system being used as a general damage detector. The optimizations conducted and results of the testing are described below.

1. Initial Test

The initial testing of the two-class detection model resulted in a recall of just above .5. We performed error analysis on the images that produced large numbers of false positives and false negatives. Visual inspection of these images revealed that picture orientation seemed to have a large effect on detection and classification. An example of image rotation having an effect on detection can be seen in Figure 35 where the same image is rotated four times resulting in four completely different detections. It is possible that the difference in detection based on orientation is the result of the training data all having the same orientation with respect to the airfield where as the images used to test the model have a different orientation. This slight difference between training and testing data can have a large impact on the accuracy of results. In an effort to maximize the accuracy of the model, a new implementation was put into place where each image loaded into the system was rotated along all four axes and analyzed individually. At the expense of increasing processing time by four times, this greatly improved the results. A possible strategy to mitigate this increase in processing time without losing classification effectiveness would be to incorporate image rotations during the training of the neural network. We leave this implementation and testing to future work.

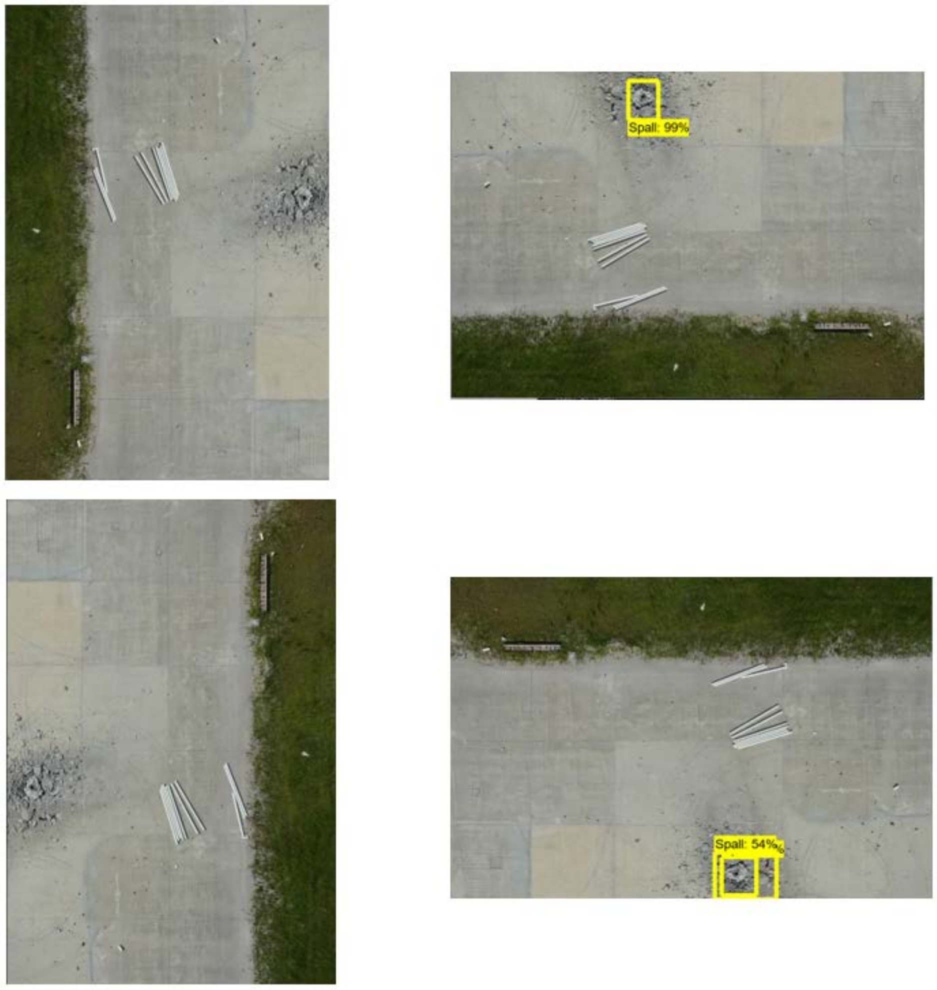


Figure 35. Example of differing detection results of the same image rotated on all four axes.

In addition to image rotation, it was noticed that sometimes the model would put multiple detections over the same area, an example of which can be seen in the bottom right picture of Figure 35. This multi-detection of the same area could also include both types of classification. Regardless, multi detection over the same area has a severe impact on results. In order to mitigate this problem, a grouping criterion was put into place where if multiple detections were over the same spot, then the detection with the highest score was used and the others were discarded. An example of this methodology can be seen in Figure 36. The results from the system testing with varying buffer size can be seen in Tables 3–6.

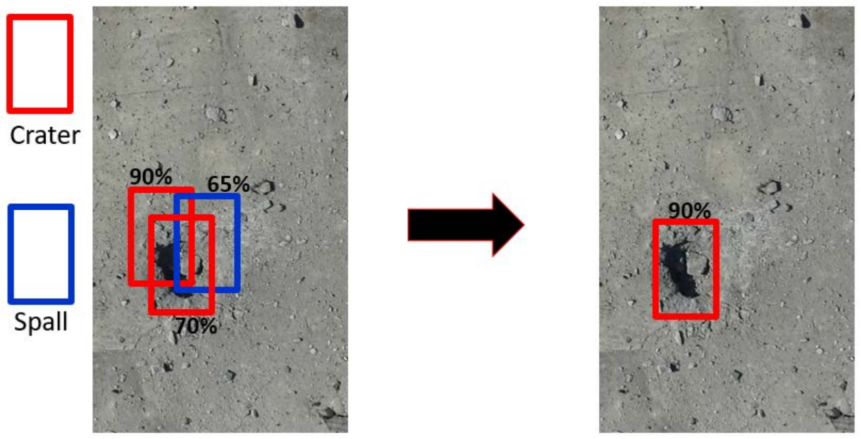


Figure 36. Example of grouping multi-detection.

Table 3. Results of two-class model with 5% error threshold in width and height.

Category	Raw Count		Analysis	Score
True Positive	407		Precision	.78
False Positive	117		Recall	.70
True Negative	1568		Accuracy	.87
False Negative	176		Negative Predictive Score	.90
			F Score	.74

Table 4. Results of two-class model with 10% error threshold in width and height.

Category	Raw Count		Analysis	Score
True Positive	426		Precision	.80
False Positive	106		Recall	.72
True Negative	1568		Accuracy	.88
False Negative	169		Negative Predictive Score	.90
			F Score	.76

Table 5. Results of two-class model with 15% error threshold in width and height.

Category	Raw Count		Analysis	Score
True Positive	442		Precision	.82
False Positive	100		Recall	.75
True Negative	1568		Accuracy	.89
False Negative	147		Negative Predictive Score	.91
			F Score	.78

Table 6. Results of two-class model with 20% error threshold in width and height.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	448		Precision	.82
False Positive	101		Recall	.76
True Negative	1568		Accuracy	.89
False Negative	141		Negative Predictive Score	.92
			F Score	.79

It is important to note that the measures of Accuracy and Negative Predictive Score look relatively high due to the ratio of test images that do not have any damaged compared to those that do have various types of damage. These results show that there is a benefit in increasing the size of the detection buffer however, that advantage is not very great after 15%.

2. Generalized Damage Detection

An analysis was done on how capable the two-class identifier was at detecting damage regardless of correct class identification. This was done by bypassing the classifier check during the pre-labeled XML comparison and instead just using the damage location to determine if the model was accurately detecting damage. Those results can be seen in Tables 7–10.

Table 7. Results of using two-class model to conduct damage detection without classification with 5% error threshold in width and height.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	499		Precision	.88
False Positive	69		Recall	.83
True Negative	1568		Accuracy	.92
False Negative	100		Negative Predictive Score	.94
			F Score	.86

Table 8. Results of using two-class model to conduct damage detection without classification with 10% error threshold in width and height.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	519		Precision	.90
False Positive	.60		Recall	.86
True Negative	1568		Accuracy	.94
False Negative	85		Negative Predictive Score	.95
			F Score	.88

Table 9. Results of using two-class model to conduct damage detection without classification with 15% error threshold in width and height.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	529		Precision	.90
False Positive	58		Recall	.87
True Negative	1568		Accuracy	.94
False Negative	77		Negative Predictive Score	.95
			F Score	.89

Table 10. Results of using two-class model to conduct damage detection without classification with 20% error threshold in width and height.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	534		Precision	.90
False Positive	58		Recall	.88
True Negative	1568		Accuracy	.94
False Negative	74		Negative Predictive Score	.96
			F Score	.89

Again, it is important to note that Accuracy and Negative Predictive Score are inflated due to the large number of images without damage compared to those with damage. Using the two-class model to conduct damage detection without classification as a measure of performance, we see better results compared to using the two-class model where we do count classification as a measure of performance. In addition, we see improvement with larger error threshold sizes that taper off after 15%. This improvement between the models is not a direct comparison since we are changing the way we score the outcome but it is worth looking at because it gives an intuition for how helpful the classifier is in a practical scenario where an operator could benefit from partial information.

3. Data Set Optimization

Similar to the procedure of the first optimization, images that produced a high False Positive or False Negative count were placed in a separate directory for further analysis in order to try and detect any trends that would lead to poor performance. A manual analysis of these images showed that many of the challenging images were only of foliage or areas with very little runway present. Two examples of images from the data set that would lead to bad detection can be seen in Figure 37. It is interesting to note that the system seemed to consistently mark large trees as craters. Images that were a vast majority of foliage were manually removed from the test set and the same tests from before were run again. Those results can be seen in Tables 11–14. The exclusion of these images is reasonable for testing effectiveness within the assumption that these issues could be resolved separately such as on the front end by ensuring an accurate UAV waypoint survey that only captures images of the runway. Additionally, we leave to future work the training the system to detect foliage and handle the images appropriately.

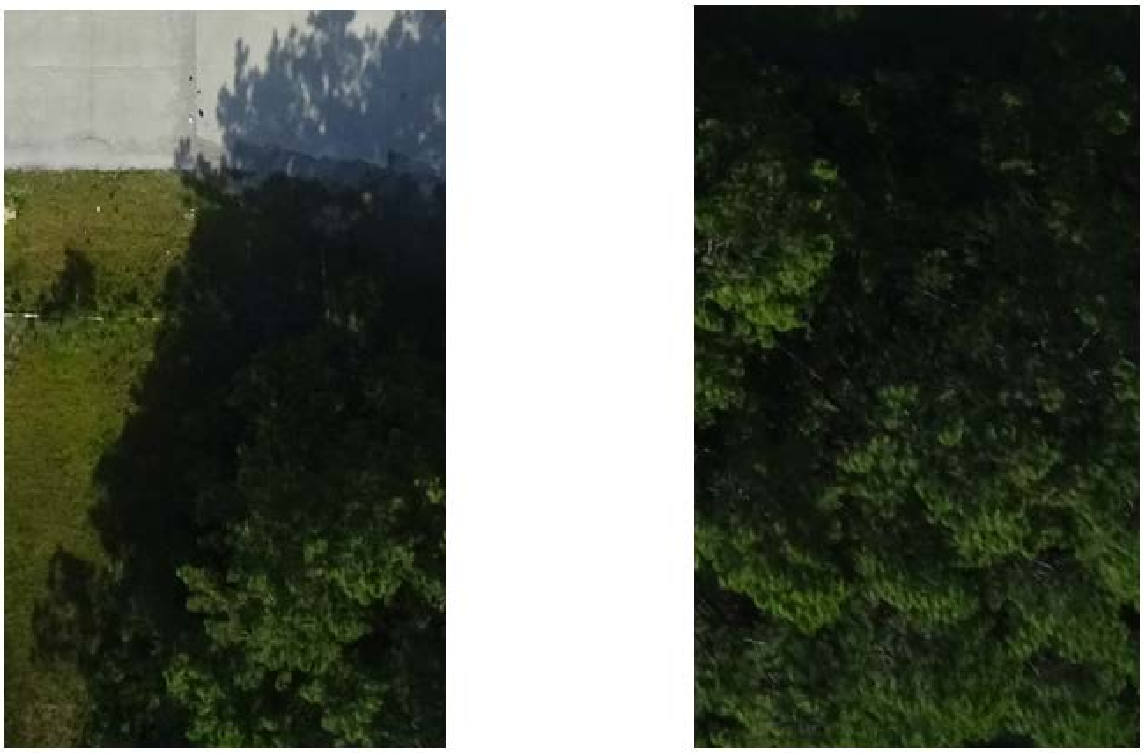


Figure 37. Examples of images from data set that lead to poor detection results.

Table 11. Results of two-class model with 5% error threshold in width and height with modified data set.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	407		Precision	.82
False Positive	91		Recall	.70
True Negative	881		Accuracy	.83
False Negative	176		Negative Predictive Score	.83
			F Score	.75

Table 12. Results of two-class model with 10% error threshold in width and height with modified data set.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	426		Precision	.82
False Positive	91		Recall	.73
True Negative	881		Accuracy	.84
False Negative	160		Negative Predictive Score	.85
			F Score	.77

Table 13. Results of two-class model with 15% error threshold in width and height with modified data set.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	442		Precision	.85
False Positive	80		Recall	.75
True Negative	881		Accuracy	.85
False Negative	147		Negative Predictive Score	.86
			F Score	.80

Table 14. Results of two-class model with 20% error threshold in width and height with modified data set.

Category	Raw Count		Analysis	Score
True Positive	448		Precision	.84
False Positive	81		Recall	.76
True Negative	881		Accuracy	.86
False Negative	141		Negative Predictive Score	.86
			F Score	.80

There is a noticeable drop in Accuracy and Negative Predictive Score between the first data set and this new pruned data set. This can be attributed to removing almost half of the images that do not have any damage in them such as images with only trees or little to no runway resulting in a much lower number of True Negatives. In addition to improved performance between the first and modified data sets, we see improved performance with a larger error threshold up to 15%.

4. Generalized Damage Detection with New Data Set

As before, the system was run using only damage detection as a measure of performance instead of both damage detection and classification. Those results can be viewed in Tables 15–18.

Table 15. Results of using two-class model to conduct damage detection without classification with 5% error threshold in width and height with modified data set.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	499		Precision	.91
False Positive	49		Recall	.83
True Negative	881		Accuracy	.90
False Negative	100		Negative Predictive Score	.90
			F Score	.87

Table 16. Results of using two-class model to conduct damage detection without classification with 10% error threshold in width and height with modified data set.

Category	**Raw Count**		**Analysis**	**Score**
True Positive	519		Precision	.92
False Positive	40		Recall	.86
True Negative	881		Accuracy	.92
False Negative	85		Negative Predictive Score	.91
			F Score	.89

Table 17. Results of using two-class model to conduct damage detection without classification with 15% error threshold in width and height with modified data set.

Category	Raw Count		Analysis	Score
True Positive	529		Precision	.93
False Positive	38		Recall	.87
True Negative	881		Accuracy	.92
False Negative	77		Negative Predictive Score	.92
			F Score	.90

Table 18. Results of using two-class model to conduct damage detection without classification with 20% error threshold in width and height with modified data set.

Category	Raw Count		Analysis	Score
True Positive	534		Precision	.93
False Positive	38		Recall	.88
True Negative	881		Accuracy	.93
False Negative	74		Negative Predictive Score	.92
			F Score	.91

When only trying to detect damage without classification on the modified data set, there is an overall improvement in performance. Again, this improvement is not meant to be a direct comparison of the systems but rather a look at practical viability for an operator to gleam partial information about the airfield damage. In addition, we see an increase in performance with higher error threshold, but this increase is very small after 15% and introduces the potential loss of precise location info.

5. Comparison of Results

Figure 38 – 42 show all of the previous results plotted against one another.

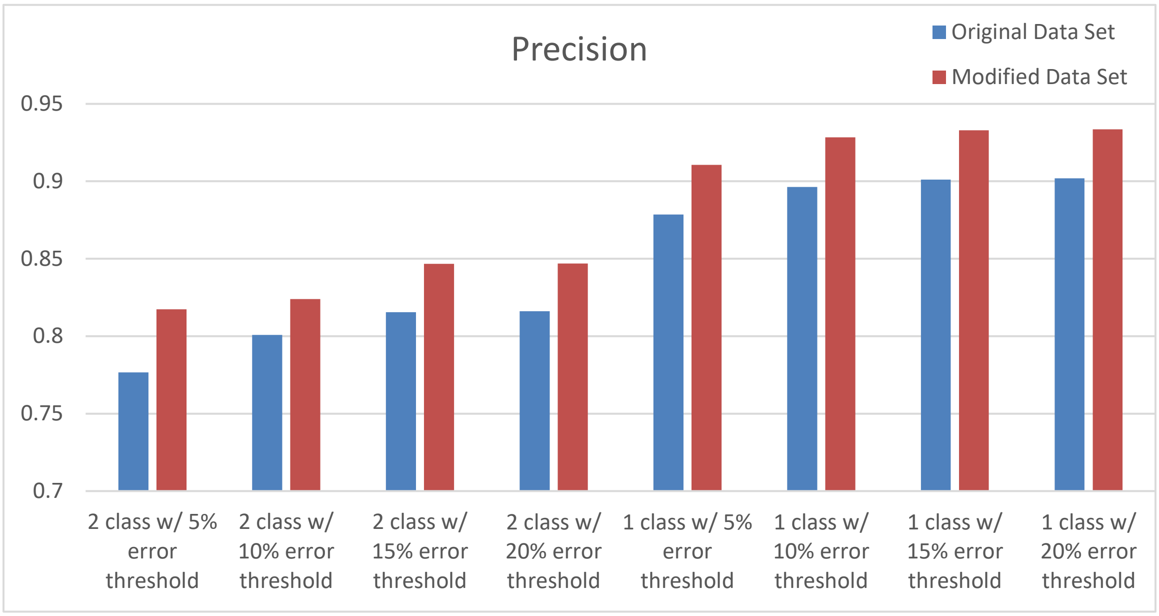


Figure 38. Comparison of Precision between two-classification and damage detection mode with original and modified data sets.

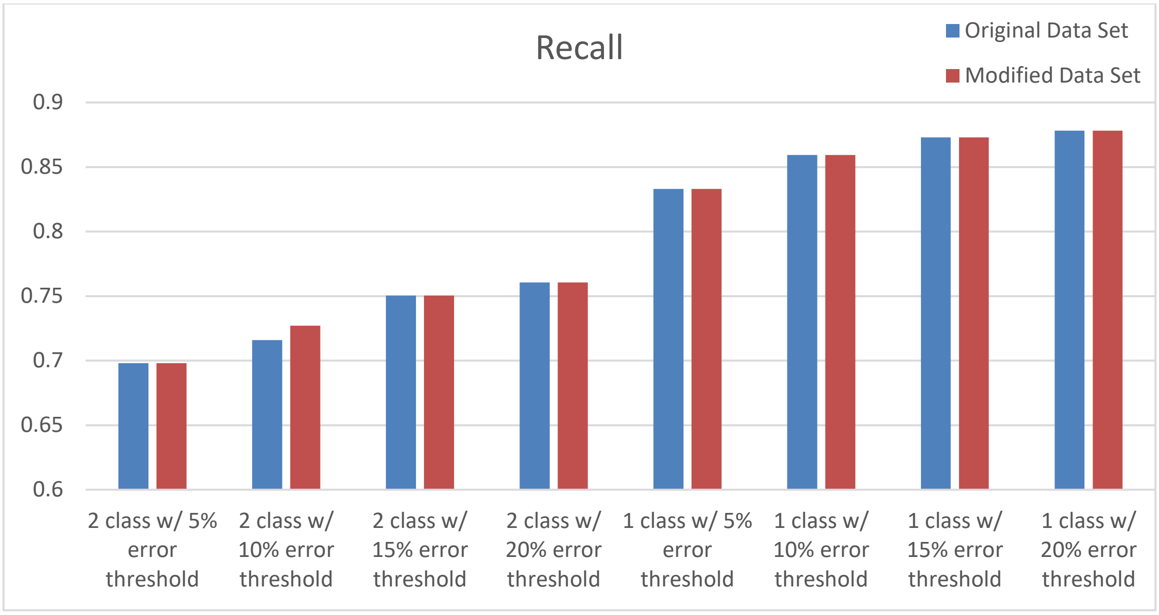


Figure 39. Comparison of Recall between two-classification and damage detection model with original and modified data sets.

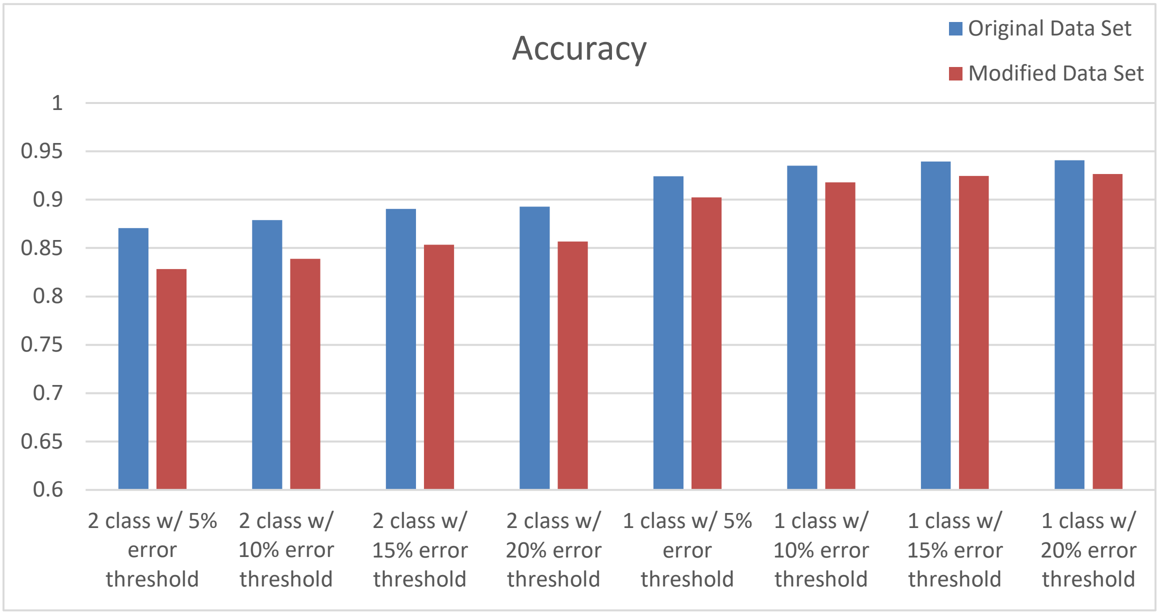


Figure 40. Comparison of Accuracy between two-classification and damage detection model with original and modified data sets.

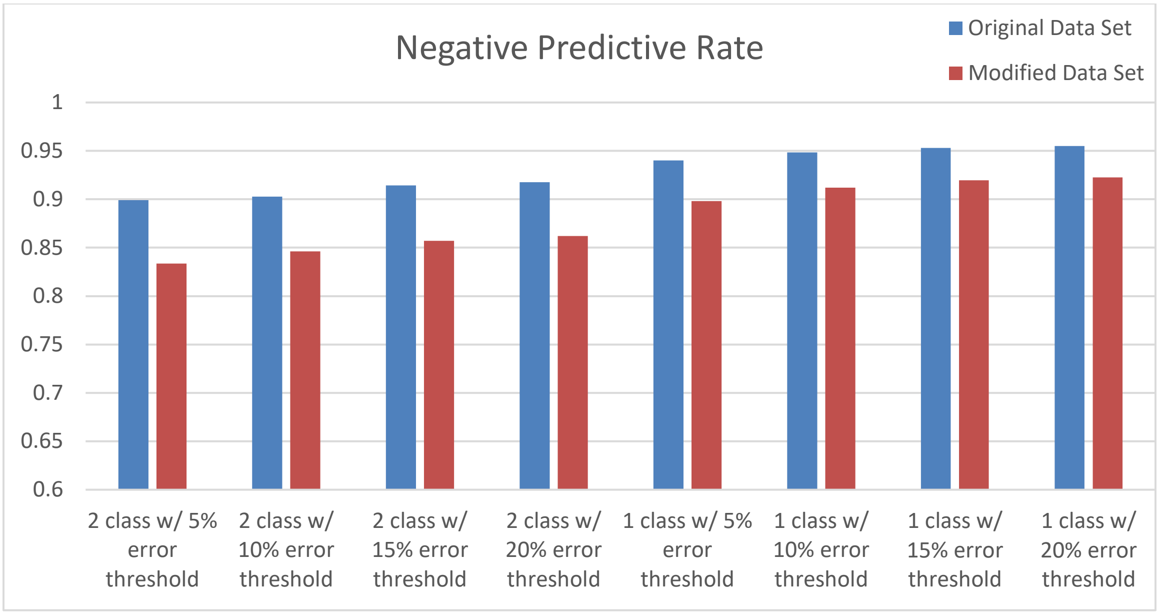


Figure 41. Comparison of Negative Predictive Rate between two-classification and damage detection model with original and modified data sets.

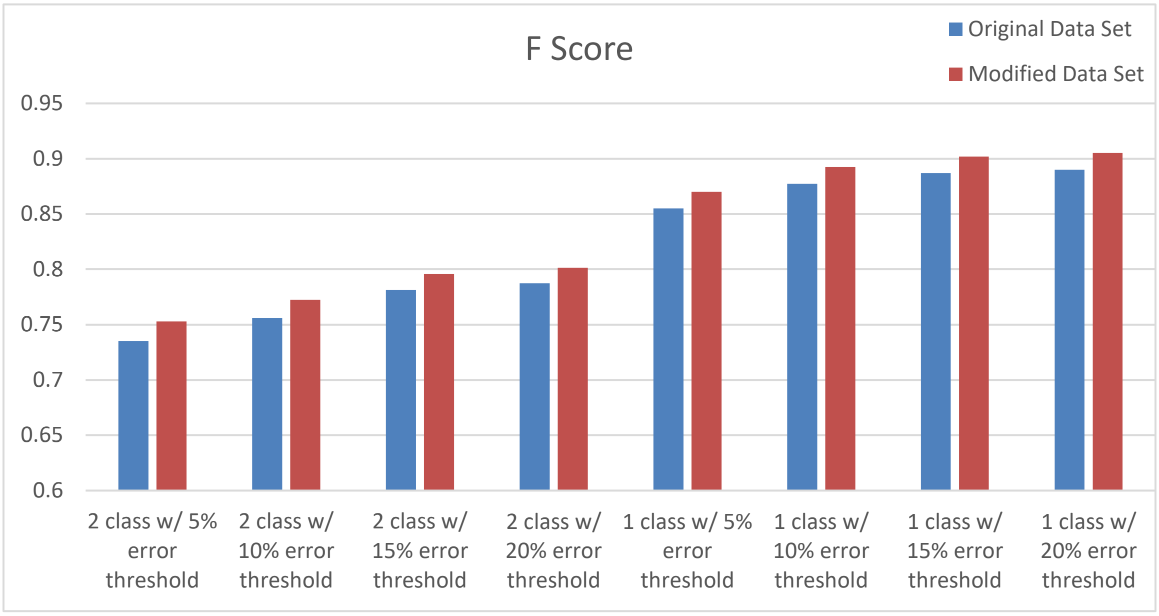


Figure 42. Comparison of Negative Predictive Rate between two-classification and damage detection model with original and modified data sets.

As discussed in earlier sections, the lower Accuracy and Negative Predictive Rate while using the model on the modified data set can be directly attributed to there being almost half as many images without damage therefore lowering the number of True Negatives. Regardless, we can see a noticeable improvement in detection performance when using the modified data set. In addition, further improvement can be leveraged by using a larger error threshold. Finally, there is a large increase in performance when using the trained two-class model as a single-class generalized damage detector. However, this comes at the expense of the system being able to identify what type of damage was detected. The comparison in metrics between the two systems is not a direct evaluation but rather a look at the potential effectiveness for practical use by an operator who could use the system to gain partial information of the airfield damage.

J. CHAPTER SUMMARY

This chapter has provided a look at the findings and results from the implementation and testing of the proposed system that was described in Chapter III. It started by listing numerous observations that impacted performance of the system including options for UAV data transfer, 3D model creation software performance, various optimizations utilizing a GPU, and mosaic creation speed. In addition, this chapter detailed the testing, optimization, and results of the neural network for automatic damage detection of spalls and craters. Chapter V summarizes these results. Presents takeaways, and lists various opportunities for further research and system enhancement.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FUTURE WORK

A. SUMMARY

The goal of this research was to develop and test a system architecture and define the associated data flow associated to conduct a fast and accurate ADA using an assortment of COTS sensors, computing hardware and software.

The system we designed and tested utilized the flexibility of a UAV combined with multiple image analysis and processing techniques to assist the EOC in MOS determination and planning of repair operations. In addition, the system was developed such that all data ingest and processing could be conducted locally. The system was broken into two distinct components: initial and secondary runway scans.

The initial runway scan consists of multiple sub components including the UAV flight over the damaged runway, automated damage detection, damage localization and mosaic creation. The goal of the initial runway scan is to provide the most critical information to the EOC as fast as possible in order to quickly make a determination if the runway is damaged, worth repairing, assist in MOS selection, and visualize, identify, and classify damage. This is accomplished by using survey mapping techniques built into Pix4Dcapture, a Phantom 4 Pro UAV and an iPad/flight controller in order to take a series of overlapping images across the runway. A trained TensorFlow neural network is then used to automatically identify and classify the damage. This information is logged for direct output and also visualized onto the image itself. Finally, the images are stitched together and grid lines are placed onto the resulting mosaic using Python's OpenCV library.

The second runway scan is meant to take place after the initial runway scan is finished and an MOS is selected. Once the area that requires repairs has been identified, the UAV can be sent back to previously logged damaged locations, use survey mapping techniques in Pix4Dcapture in order to collect more images of the area of interest. These images can then be used to create 3D models in programs such as WebODM and Agisoft

PhotoScan. Finally, the 3D models can be incorporated into existing modeling software such as GeoExPT/AutoCAD in order to estimate required repair materials.

B. CONCLUSION

Our system offers a cost-effective and low-training-requirement design that is capable of assisting the EOC in conducting ADA. The system was broken down into multiple sub elements including data collection, mosaic creation, automatic damage detection, 3D model creation, and further repair planning analysis. A viable set of tools and software was identified and tested resulting in a data flow that produces a mosaic of an airfield, overlays grid lines onto the mosaic, automatically identifies and logs the position of spalls and craters, and generates 3D models for further analysis. These products can have an immediate and direct positive impact on the EOC's ability to conduct ADA by increasing the accuracy and decreasing the time to conduct the assessment. In addition, this system lays the groundwork for continued development and integration with multiple sensors and pieces of technology to create an even more capable solution for automated ADA.

Our system has many draw-backs and limitations such as reliance on a UAV. Extreme weather conditions such as high winds, fog, and rain can directly impact the capability of using a UAV as a source of data collection. In addition, the application of the sub elements was not based on exhaustive testing but rather were used to prove applicability and capability in the data flow of the overall system. Further areas of possible improvement or system enhancement are detailed below.

C. FUTURE WORK

Many of the individual components of our system design may benefit from further testing, evaluation and fine-tuning. Regarding placement of the grid lines after the mosaic is created, detailed analysis could be conducted on how accurate the grid lines are based on separate GPS readings for of the testing site. In addition, the system would benefit from software-based error correction and handing if the mosaic is tilted in order to maximize grid line accuracy.

Further experimentation on the optimal drone speed while taking images could have a large effect on the speed of the data collection. Instead of stopping at each waypoint to take an image, it may be possible to orient the camera angle and modify the camera parameters based on the speed of the UAV so that data could be collected faster without sacrificing accuracy or creating distorted images. During mosaic creation, different image stitching techniques could be tested in order to maximize speed of the image stitch and also determine optimum image percent overlap decreasing time required for data collection.

Vast amounts of testing could be done on the TensorFlow neural network model including testing different models, varying the training data set, including image rotations in the training data set, using a larger dataset, training the system with more classes, utilizing transfer learning with pre-trained models on large data sets, and modifying training parameters in order to maximize detection capabilities. In addition, incorporation of UXO and camouflet detection are a much-needed requirement for the ADA problem set. Testing and evaluation on the effects of detection at different altitudes and lighting, and the accuracy of both UAV GPS and detected damage locations remain areas for improvement. Finally, with the continued advancement of COTS UAVs, conducting real-time damage detection on board the UAV may decrease overall evaluation time.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. VIRTUAL ENVIRONMENT SET UP

The following packages were installed on the Anaconda virtual environment in order to conduct TensorFlow model training and use: absl-py 0.2.2, astor 0.6.2, backcall 0.1.0, bleach 1.5.0, certify 2018.4.16, chardet 3.0.4, colorama 0.3.9, cycler 0.10.0, Cython 0.28.0, decorator 4.3.0, Django 2.0.5, entrypoints 0.2.3, gast 0.2.0, geographiclib 1.49, geopy 1.14.0, grpcio 1.12.0, h5py 2.7.1, html5lib .9999999, idna 2.6, image 1.5.24, imutils 0.4.6, ipykernel 4.8.2, ipython 6.4.0, ipython-genutils 0.2.0, ipywidgets 7.2.1, jedi 0.12.0, Jinja2 2.10, jsonschema 2.6.0, jupyter 1.0.0, jupyter-client 5.2.3, jupyter-console 5.2.0, jupyter-core 4.4.0, Keras 2.1.6, kiwisolver 1.0.1, lxml 4.2.1, Markdown 2.6.11, MarkupSafe 1.0, matplotlib 2.2.2, mistune 0.8.3, nbconvert 5.3.1, nbformat 4.4.0, notebook 5.5.0, numpy 1.14.3, opencv-contrib-python 3.4.1.15, pandas 0.23.0, pandas-datareader 0.6.0, pandocfilters 1.4.2, parso 0.2.1, pickleshare 0.7.4, Pillow 5.1.0, pip 10.0.1, prompt-toolkit 1.0.15, protobuf 3.5.2.post1, Pygments 2.2.0, pyparsing 2.2.0, python-dateutil 2.7.3, pytz 2018.4, pywinpty 0.5.3, PyYAML 3.12, pyzm1 17.0.0, qtconsole 4.3.1, requests 2.18.4, requests-file 1.4.3, requests-ftp 0.3.1, scikit-learn 0.19.1, scipy 1.1.0, Send2Trash 1.5.0, setuptools 39.1.0, simplegeneric 0.8.1, six 1.11.0, sklearn 0.0, tensorboard 1.9.0, tensorflow-gpu 1.9.0/tensorflow 1.5.0, termcolor 1.1.0, terminado 0.8.1, testpath 0.3.1, tornado 5.0.1, traitlets 4.3.2, urllib3 1.22, wcwidth 0.1.7, Werkzeug 0.14.1, wheel 0.31.1, widgetsnbextension 3.2.1, win-unicode-console 0.5, wincertstore 0.2, wrapt 1.10.11.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Anaconda [Computer software]. (2018). Retrieved from https://www.anaconda.com/

Angelova, A., Krizhevsky, A.,& Vanhoucke, V. (2015). Pedestrian detection with a large-field-of-view deep network. *International Conference on Robotics and Automation, Proceedings of ICRA 2015.* Retrieved from https://storage.googleapis.com/pub-tools-public-publication-data/pdf/43849.pdf

Copeland, M. (2016, July 29). What's the difference between artificial intelligence, machine learning, and deep learning? [Blog post]. Retrieved from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Cuingnet, R. (2016, May 20). Re: Rotate and image without cropping in OpenCV in C++. [Blog comment]. Retrieved from https://stackoverflow.com/questions/22041699/rotate-an-image-without-cropping-in-opencv-in-c/37347070#37347070

Department of the Air Force. (2012, December 17). *Air Force Pamphlet (AFPAM) 10-219*, Volume 4. Washington, DC*: Airfield Damage Repair Operations.*

GeoExPT [Computer software]. (2018). Retrieved from http://www.geoexpt.com/

George, W. (2018, May 02). *Agisoft PhotoScan 1.4.1—Multi GPU scaling*. Retrieved from https://www.pugetsystems.com/labs/articles/Agisoft-PhotoScan-1-4-1---Multi-GPU-Scaling-1147/

Hulstaert, L. (2018, April 19). A beginner's guide to object detection. DataCamp. Retrieved from https://www.datacamp.com/community/tutorials/object-detection-guide

Kim, H., Lee, J., Ahn, E., Cho, S., Shin, M., & Sim, S.-H. (2017). Concrete crack identification using a UAV incorporating hybrid image processing. *Sensors, 17(9)*, 2052. doi: 10.3390/s17092052

iFerret. (2015). Retrieved from http://www.thestratechgroup.com/iv_iferret.asp

iFerret Hydra. (2015). Retrieved from http://www.thestratechgroup.com/iv_iferret_hydra.asp

iFerret Mobile. (2015). Retrieved from http://www.thestratechgroup.com/iv_iferret_mobile.asp

ImageNet. (2016). Retrieved from http://www.image-net.org/

Christiansen, M. P., Laursen, M. S., Jørgensen, R. N., Skovsen, S., & Gislum, R. (2017). Designing and testing a UAV mapping system for agriculture field surveying. *Sensors*, *17*, 2703. doi: 10.3390/s17122703

maxbellec. (2018). Get latitude and longitude from EXIF using PIL [Program code]. Retrieved from https://gist.github.com/maxbellec/dbb60d136565e3c4b805931f5aad2c6d

Nielsen, M. A. (2015). Neural networks and deep learning. Determination Press. Retieved from http://neuralnetworksanddeeplearning.com/index.html

OpenDroneMap [Computer program]. (2018). Retrieved from https://github.com/OpenDroneMap/OpenDroneMap

Phantom 4. (2018). Retrieved from https://www.dji.com/phantom-4/info

PhotoScan. (2018). Agisoft [Computer program]. Retrieved from http://www.agisoft.com/

Pix4Dcapture [Mobile application program]. (n.d.). Retrieved from https://pix4d.com/product/pix4dcapture/

Pothole Synset. (2010). Retrieved from http://image-net.org/synset?wnid=n09398076

Schenk, T. (2005). *Introduction to photogrammetry*. Columbus: Ohio State University. Retreived from http://www.mat.uc.pt/~gil/downloads/IntroPhoto.pdf

Siebert, S., & Teizer, J. (2014). Mobile 3D mapping for surveying earthwork using an Unmanned Aerial Vehicle (UAV). *Automation in Construction*, *41*, 1-14. doi: 10.1016/j.autcon.2014.01.004

Smith, G. (n.d.). Digitial orthophotography and GIS. Retrieved from http://proceedings.esri.com/library/userconf/proc95/to150/p124.html

Super BullsEye II. (2015). Retrieved from http://www.thestratechgroup.com/iv_bullseye.asp

Szeliski, R. (2005). *Image alignment and stitching: A tutorial*. Redmond: Microsoft Research. Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2004/10/tr-2004-92.pdf

Tran, D. (2017). *Raccoon detector dataset* [Dataset]. Retrieved from https://github.com/datitran/raccoon_dataset

What is a container. (2018). Retrieved from https://www.docker.com/resources/what-container

What is ground sample distance (GSD) and how does it affect your drone data? (2018, February 28). Retrieved from https://www.propelleraero.com/blog/ground-sample-distance-gsd-calculate-drone-data/

Zhang, C., & Elaksher, A. (2010). 3D Reconstruction from UAV-acquired Imagery for Road Surface Distress Assessment. *31st Asian Conference on Remote Sensing* 2010, *ACRS 2010. 1.*

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST